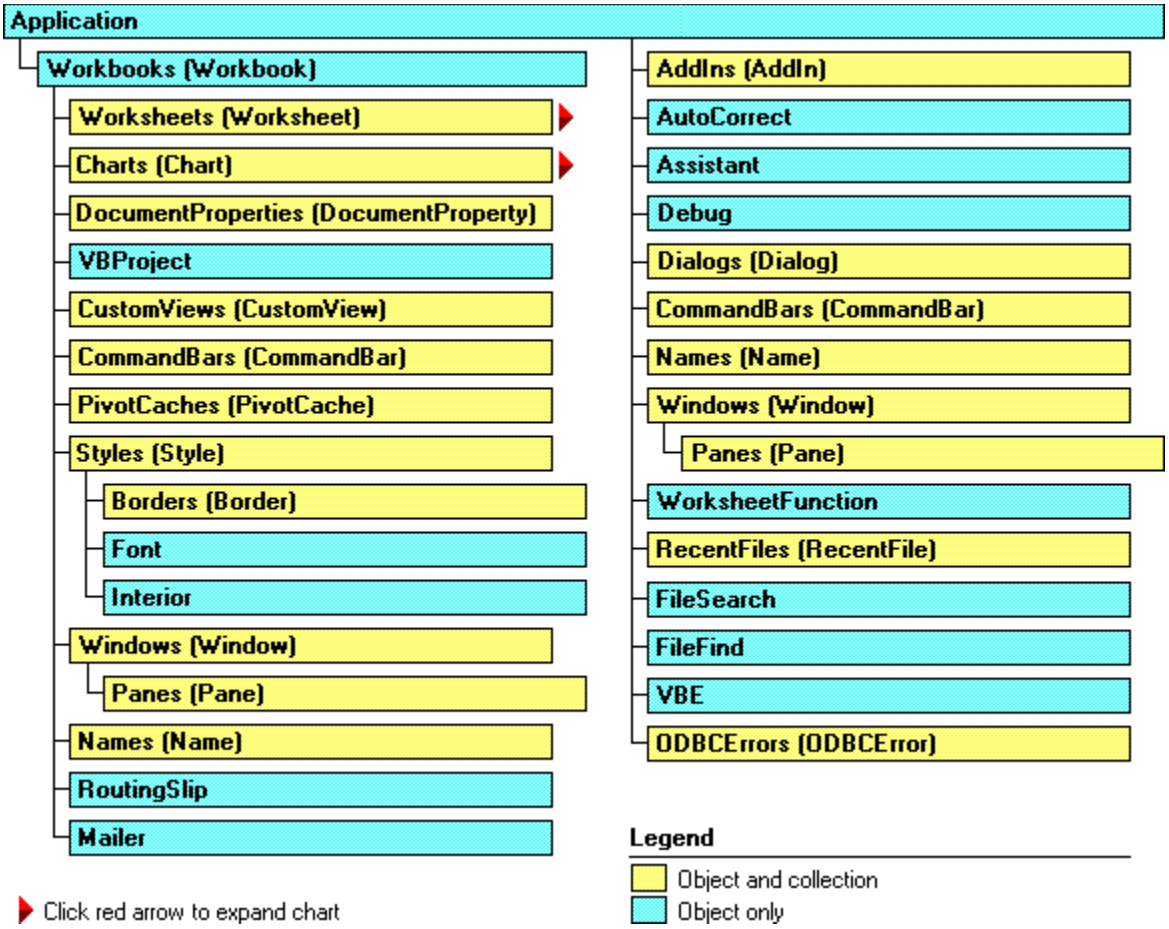
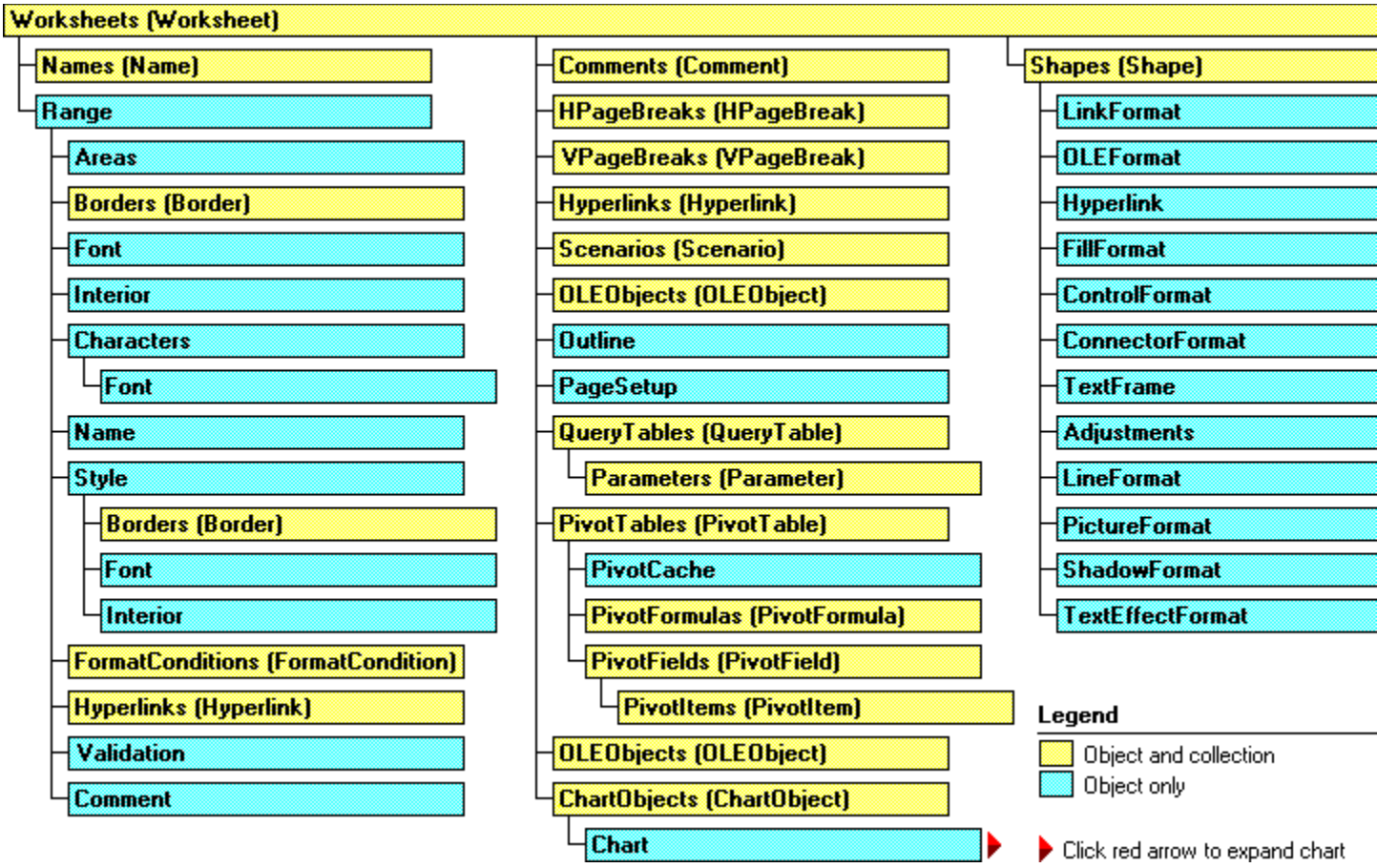
Microsoft Excel Objects

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xltocObjectModelApplicationC "}



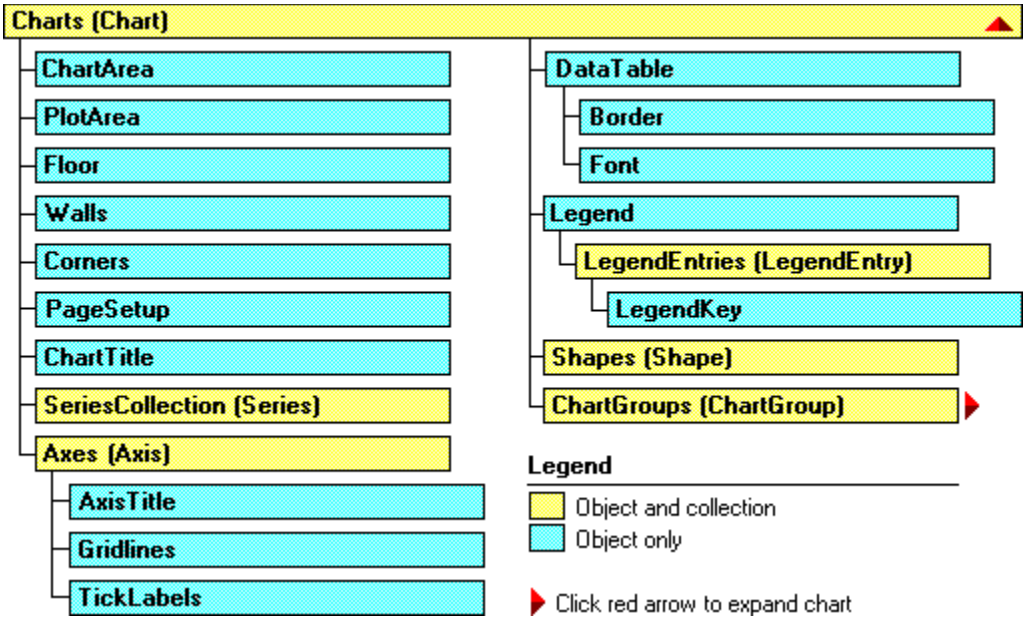
Microsoft Excel Objects (Worksheet)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xltocObjectModelWorksheetC "}



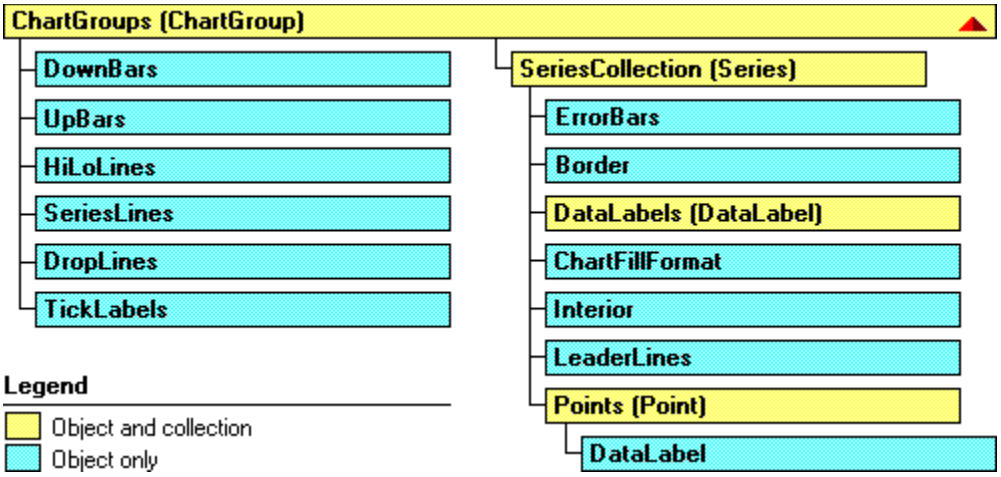
Microsoft Excel Objects (Charts)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xltocObjectModelChartsC "}



Microsoft Excel Objects (ChartGroups)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xltocObjectModelChartGroupsC "}



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 Excel**, and then click **Change Option**.
3. Click **Help and sample files**, and then click **Change Option**.
4. Make sure that the **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 Excel**, and then click **Change Option**.
3. Click **Help and sample files**, and then click **Change Option**.
4. Make sure that the **Help for Visual Basic** check box is selected.
5. Continue with Setup

Help Topic Not Available

The Help topic cannot be displayed because Microsoft Excel Help cannot be found or was not installed.

To install Microsoft Excel Help

1. Run Microsoft Office 97 Setup, and click **Add/Remove**.
2. Click **Microsoft Excel**, and then click **Change Option**.
3. Click **Help and sample files**, and then click **Change Option**.
4. Make sure that the **Help for Microsoft Excel** check box is selected.
5. Continue with Setup

Help Topic Not Available

The Help topic cannot be displayed because Data Access Objects Help cannot be found or was not installed.

To install Data Access Objects Help

1. Run Microsoft Office 97 Setup, and click **Add/Remove**.
2. Click **Microsoft Access**, and then click **Change Option**.
3. Click **Help**, and then click **Change Option**.
4. Make sure that the **Language Reference** check box is selected.
5. Continue with Setup

Changes to the Microsoft Excel 97 Object Model

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscObjectModelChangesC"}

Extensive changes have been made to the Microsoft Excel 97 Visual Basic object model to support new and improved features in the application. Many objects, properties, and methods have been replaced. To provide backward compatibility, most of the replaced components have been hidden rather than removed. This means that they don't show up in the object browser by default, but old code that uses the hidden components will still work correctly without modification. When you write new code, however, you should use the new objects, properties, and methods.

The major feature changes made for Visual Basic in Microsoft Excel 97 are listed in the following table.

Feature	Description
Shapes	Replaces the drawing layer (Arc, Oval, Line, and so on) with a consistent and improved object model in all Microsoft Office applications.
UserForms, **ActiveX** controls	Provides a consistent and expandable control and dialog box interface in all Microsoft Office applications.
CommandBars	Provides a consistent and expandable menu and toolbar interface in all Microsoft Office applications.

For more information, see one of the following topics:

New Objects

New Properties and Methods (by Object)

New Properties and Methods (Alphabetic List)

Hidden Objects

Hidden Properties and Methods

Methods with New Arguments

Changes to Visual Basic in Microsoft Excel for Windows 95

Hidden Objects

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscHiddenObjectsC"}

Objects that have been hidden in the Microsoft Excel 97 Visual Basic object model are listed in the following table. These objects are supported only for backward compatibility; for new code, you should use the replacement functionality provided in Microsoft Excel 97. To view hidden objects in the Object Browser, right-click in the Object Browser window and click **Show Hidden Members** on the shortcut menu. For more information about the changes to the Microsoft Excel 97 object model, see one of the following topics:

New Objects

New Properties and Methods (by Object)

New Properties and Methods (Alphabetic List)

Hidden Properties and Methods

Methods with New Arguments

Changes to Visual Basic in Microsoft Excel for Windows 95

Hidden objects	Replacement
Arc, **Arcs**, **Drawing**, **DrawingObjects**, **Drawings**, **Label**, **Labels**, **Line**, **Lines**, **Oval**, **Ovals**, **Picture**, **Pictures**, **Rectangle**, **Rectangles**	New **Shapes** drawing layer
Button, **Buttons**, **CheckBox**, **CheckBoxes**, **DialogFrame**, **DropDown**, **DropDowns**, **EditBox**, **EditBoxes**, **GroupBox**, **GroupBoxes**, **GroupObject**, **GroupObjects**, **ListBox**, **ListBoxes**, **OptionButton**, **OptionButtons**, **ScrollBar**, **ScrollBars**, **Spinner**, **Spinners**, **TextBox**, **TextBoxes**	**ActiveX** controls
Menu, **MenuBar**, **MenuBars**, **MenuItem**, **MenuItems**, **Menus**, **Toolbar**, **ToolbarButton**, **ToolbarButtons**, **Toolbars**	**CommandBars**
Module, **Modules**	VBE extensibility object model
DialogSheet, **DialogSheets**	**Custom Forms**

New Objects

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscNewObjectsC"}

Objects that were added to Visual Basic in Microsoft Excel 97 are listed in the following table. For more information about the changes to the Microsoft Excel 97 object model, see one of the following topics:

New Properties and Methods (by Object)

New Properties and Methods (Alphabetic List)

Hidden Objects

Hidden Properties and Methods

Methods with New Arguments

Changes to Visual Basic in Microsoft Excel for Windows 95

Object	Description
CalculatedFields, **CalculatedItems**	New PivotTable functionality
Comment, **Comments**	New comment functionality; replaces cell notes
ControlFormat, **LinkFormat**, **OLEFormat**	Expose old Microsoft Excel controls and OLE objects in the **Shapes** collection
CustomView, **CustomViews**	New custom view feature
DataTable	Chart data table
ChartFillFormat, **FillFormat**	Chart and shape fill formatting
FormatCondition, **FormatConditions**	New conditional format feature
HPageBreak, **HPageBreaks**, **VPageBreak**, **VPageBreaks**	New horizontal and vertical page break features
Hyperlink, HyperLinks	Range and shape hyperlinks
LeaderLines	Chart leader lines that connect data labels to points
ODBCError, **ODBCErrors**, **Parameter**, **Parameters**, **QueryTable**, **QueryTables,**	New parameterized query features
PivotCache, PivotCaches, **PivotFormula**, **PivotFormulas**	New PivotTable features
RecentFile, **RecentFiles**	List of recently used files
Adjustments, **CalloutFormat**, **ChartColorFormat**, **ColorFormat**, **ConnectorFormat**, **FreeformBuilder**, **GroupShapes**, **LineFormat**, **PictureFormat**, **ShadowFormat**, **Shape**, **ShapeNode**, **ShapeNodes**, **Shapes**, **TextEfectFormat**, **TextFrame**, **ThreeDFormat**	New drawing layer and chart fill
Validation	Range data validation

New Properties and Methods (by Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscNewPMbyObjC"}

Properties and methods that have been added to existing objects in Microsoft Excel 97 are listed in the following table (sorted by object name). For more information about the changes to the Microsoft Excel 97 object model, see one of the following topics:

New Objects

New Properties and Methods (Alphabetic List)

Hidden Objects

Hidden Properties and Methods

Methods with New Arguments

Changes to Visual Basic in Microsoft Excel for Windows 95

Object	New properties and methods
Application	**Assistant** Property, **CommandBars** Property, **ControlCharacters** Property, **CursorMovement** Property, **DefaultSaveFormat** Property, **DefaultSheetDirection** Property, **DisplayCommentIndicator** Property, **EnableEvents** Property, **EnableSound** Property, **FileFind** Property, **FileSearch** Property, **ODBCErrors** Property, **ODBCTimeout** Property, **PivotTableSelection** Property, **RecentFiles** Property, **RollZoom** Property, **SaveWorkspace** Method, **ShowChartTipNames** Property, **ShowChartTipValues** Property, **UILanguage** Property, **UserControl** Property, **VBE** Property, **WorksheetFunction** Property
AutoCorrect	**CorrectCapsLock** Property, **CorrectSentenceCap** Property
Axis	**BaseUnit** Property, **BaseUnitIsAuto** Property, **CategoryType** Property, **Height** Property, **Left** Property, **MajorUnitScale** Property, **MinorUnitScale** Property, **Top** Property, **Width** Property
AxisTitle	**AutoScaleFont** Property, **Fill** Property, **ReadingOrder** Property
Chart	**ApplyCustomType** Method, **BarShape** Property, **ChartType** Property, **CodeName** Property, **DataTable** Property, **Export** Method, **GetChartElement** Method, **HasDataTable** Property, **Hyperlinks** Property, **Location** Method, **PlotBy** Property, **ProtectData** Property, **ProtectFormatting** Property, **ProtectGoalSeek** Property, **ProtectSelection** Property, **Refresh** Method, **SetSourceData** Method, **Shapes** Property, **ShowWindow** Property
ChartArea	**AutoScaleFont** Property, **Fill** Property
ChartGroup	**BubbleScale** Property, **Has3DShading** Property, **Index** Property, **SecondPlotSize** Property, **ShowNegativeBubbles** Property, **SizeRepresents** Property, **SplitType** Property, **SplitValue** Property
ChartObject	**ProtectChartObject** Property, **ShapeRange** Property
ChartObjects	**ShapeRange** Property
Charts	**HPageBreaks** Property, **VPageBreaks** Property
ChartTitle	**AutoScaleFont** Property, **Fill** Property, **ReadingOrder** Property
DataLabel	**AutoScaleFont** Property, **Fill** Property, **Position** Property, **ReadingOrder** Property
DataLabels	**AutoScaleFont** Property, **Fill** Property, **Position** Property, **ReadingOrder** Property
DownBars	**Fill** Property

Floor	**Fill** Property, **Paste** Method, **PictureType** Property
Interior	**InvertIfNegative** Property
Legend	**AutoScaleFont** Property, **Clear** Method, **Fill** Property
LegendEntry	**AutoScaleFont** Property, **Height** Property, **Left** Property, **Top** Property, **Width** Property
LegendKey	**Fill** Property, **Height** Property, **Left** Property, **MarkerSize** Property, **PictureType** Property, **PictureUnit** Property, **Shadow** Property, **Top** Property, **Width** Property
OLEObject	**LinkedCell** Property, **ListFillRange** Property, **ProgId** Property, **ShapeRange** Property, **SourceName** Property
OLEObjects	**AutoLoad** Property, **ShapeRange** Property, **SourceName** Property
PageSetup	**PrintComments** Property
PivotField	**AutoShow** Method, **AutoShowCount** Property, **AutoShowField** Property, **AutoShowRange** Property, **AutoShowType** Property, **AutoSort** Method, **AutoSortField** Property, **AutoSortOrder** Property, **CalculatedItems** Method, **Delete** Method, **DragToColumn** Property, **DragToHide** Property, **DragToPage** Property, **DragToRow** Property, **Formula** Property, **IsCalculated** Property, **MemoryUsed** Property, **ServerBased** Property, **ShowAllItems** Property
PivotItem	**Delete** Method, **Formula** Property, **IsCalculated** Property, **RecordCount** Property
PivotItems	**Add** Method
PivotTable	**CacheIndex** Property, **CalculatedFields** Method, **DisplayErrorString** Property, **DisplayNullString** Property, **EnableDrilldown** Property, **EnableFieldDialog** Property, **EnableWizard** Property, **ErrorString** Property, **GetData** Method, **ListFormulas** Method, **ManualUpdate** Property, **MergeLabels** Property, **NullString** Property, **PageFieldOrder** Property, **PageFieldStyle** Property, **PageFieldWrapCount** Property, **PageRangeCells** Property, **PivotCache** Method, **PivotFormulas** Method, **PivotSelect** Method, **PivotSelection** Property, **PivotTableWizard** Method, **PreserveFormatting** Property, **SelectionMode** Property, **SubtotalHiddenPageItems** Property, **TableStyle** Property, **Tag** Property, **Update** Method, **VacatedStyle** Property
PlotArea	**Fill** Property, **InsideHeight** Property, **InsideLeft** Property, **InsideTop** Property, **InsideWidth** Property
Point	**ApplyPictToEnd** Property, **ApplyPictToFront** Property, **ApplyPictToSides** Property, **Fill** Property, **MarkerSize** Property, **SecondaryPlot** Property, **Shadow** Property
Range	**AddComment** Method, **ClearComments** Method, **Comment** Property, **FormatConditions** Property, **FormulaLabel** Property, **Hyperlinks** Property, **IndentLevel** Property, **InsertIndent** Method, **Merge** Method, **MergeArea** Property, **MergeCells** Property, **QueryTable** Property, **ReadingOrder** Property, **ShrinkToFit** Property, **UnMerge** Method, **Validation** Property, **Value2** Property
Series	**ApplyCustomType** Method, **ApplyPictToEnd** Property, **ApplyPictToFront** Property, **ApplyPictToSides** Property, **BarShape** Property, **BubbleSizes** Property, **ChartType** Property, **Fill** Property, **Has3DEffect** Property, **HasLeaderLines** Property, **LeaderLines** Property, **MarkerSize** Property, **Shadow** Property

SeriesCollection	**NewSeries** Method
Sheets	**HPageBreaks** Property, **VPageBreaks** Property
Style	**BuiltIn** Property, **IndentLevel** Property, **MergeCells** Property, **ReadingOrder** Property, **ShrinkToFit** Property
TickLabels	**AutoScaleFont** Property, **ReadingOrder** Property
UpBars	**Fill** Property
Walls	**Fill** Property, **Paste** Method, **PictureType** Property, **PictureUnit** Property
Window	**EnableResize** Property, **View** Property
Workbook	**AcceptAllChanges** Method, **AcceptLabelsInFormulas** Property, **AddToFavorites** Method, **AutoUpdateFrequency** Property, **AutoUpdateSaveChanges** Property, **ChangeHistoryDuration** Property, **CodeName** Property, **CommandBars** Property, **ConflictResolution** Property, **CustomViews** Property, **FollowHyperlink** Method, **HighlightChangesOnScreen** Property, **HighlightChangesOptions** Method, **IsAddin** Property, **KeepChangeHistory** Property, **ListChangesOnNewSheet** Property, **MergeWorkbook** Method, **PersonalViewListSettings** Property, **PersonalViewPrintSettings** Property, **PivotCaches** Method, **ProtectSharing** Method, **PurgeChangeHistoryNow** Method, **RefreshAll** Method, **RejectAllChanges** Method, **Reload** Method, **RemoveUser** Method, **ResetColors** Method, **TemplateRemoveExtData** Property, **UnprotectSharing** Method, **UserControl** Property, **VBProject** Property
Worksheet	**CircleInvalid** Method, **ClearCircles** Method, **CodeName** Property, **Comments** Property, **DisplayPageBreaks** Property, **DisplayRightToLeft** Property, **EnableCalculation** Property, **EnableSelection** Property, **HPageBreaks** Property, **Hyperlinks** Property, **QueryTables** Property, **ResetAllPageBreaks** Method, **ScrollArea** Property, **Shapes** Property, **VPageBreaks** Property
Worksheets	**HPageBreaks** Property, **VPageBreaks** Property

New Properties and Methods (Alphabetic List)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscNewPMAlphaC"}

Properties and methods that have been added to existing objects in Microsoft Excel 97 are listed in the following table (sorted alphabetically by property or method name). For more information about the changes to the Microsoft Excel 97 object model, see one of the following topics:

New Objects

New Properties and Methods (by Object)

Hidden Objects

Hidden Properties and Methods

Methods with New Arguments

Changes to Visual Basic in Microsoft Excel for Windows 95

New property or method	Objects
AcceptAllChanges Method	**Workbook**
AcceptLabelsInFormulas Property	**Workbook**
Add Method	**PivotItems**
AddComment Method	**Range**
AddToFavorites Method	**Workbook**
ApplyCustomType Method	**Chart**, **Series**
ApplyPictToEnd Property	**Point**, **Series**
ApplyPictToFront Property	**Point**, **Series**
ApplyPictToSides Property	**Point**, **Series**
Assistant Property	**Application**
AutoLoad Property	**OLEObjects**
AutoScaleFont Property	**AxisTitle**, **ChartArea**, **ChartTitle**, **DataLabel**, **DataLabels**, **Legend**, **LegendEntry**, **TickLabels**
AutoShow Method	**PivotField**
AutoShowCount Property	**PivotField**
AutoShowField Property	**PivotField**
AutoShowRange Property	**PivotField**
AutoShowType Property	**PivotField**
AutoSort Method	**PivotField**
AutoSortField Property	**PivotField**
AutoSortOrder Property	**PivotField**
AutoUpdateFrequency Property	**Workbook**
AutoUpdateSaveChanges Property	**Workbook**
BarShape Property	**Chart**, **Series**
BaseUnit Property	**Axis**
BaseUnitIsAuto Property	**Axis**
BubbleScale Property	**ChartGroup**
BubbleSizes Property	**Series**
BuiltIn Property	**Style**
CacheIndex Property	**PivotTable**

CalculatedFields Method	**PivotTable**
CalculatedItems Method	**PivotField**
CategoryType Property	**Axis**
ChangeHistoryDuration Property	**Workbook**
ChartType Property	**Chart**, **Series**
CircleInvalid Method	**Worksheet**
Clear Method	**Legend**
ClearCircles Method	**Worksheet**
ClearComments Method	**Range**
CodeName Property	**Chart**, **Workbook**, **Worksheet**
CommandBars Property	**Application**, **Workbook**
Comment Property	**Range**
Comments Property	**Worksheet**
ConflictResolution Property	**Workbook**
ControlCharacters Property	**Application**
CorrectCapsLock Property	**AutoCorrect**
CorrectSentenceCap Property	**AutoCorrect**
CursorMovement Property	**Application**
CustomViews Property	**Workbook**
DataTable Property	**Chart**
DefaultSaveFormat Property	**Application**
DefaultSheetDirection Property	**Application**
Delete Method	**PivotField**, **PivotItem**
DisplayCommentIndicator Property	**Application**
DisplayErrorString Property	**PivotTable**
DisplayNullString Property	**PivotTable**
DisplayPageBreaks Property	**Worksheet**
DisplayRightToLeft Property	**Worksheet**
DragToColumn Property	**PivotField**
DragToHide Property	**PivotField**
DragToPage Property	**PivotField**
DragToRow Property	**PivotField**
EnableCalculation Property	**Worksheet**
EnableDrilldown Property	**PivotTable**
EnableEvents Property	**Application**
EnableFieldDialog Property	**PivotTable**
EnableResize Property	**Window**
EnableSelection Property	**Worksheet**
EnableSound Property	**Application**
EnableWizard Property	**PivotTable**
ErrorString Property	**PivotTable**
Export Method	**Chart**
FileFind Property	**Application**
FileSearch Property	**Application**
Fill Property	**AxisTitle**, **ChartArea**, **ChartTitle**, **DataLabel**,

	DataLabels, **DownBars**, **Floor**, **Legend**, **LegendKey**, **PlotArea**, **Point**, **Series**, **UpBars**, **Walls**
FollowHyperlink Method	**Workbook**
FormatConditions Property	**Range**
Formula Property	**PivotField**, **PivotItem**
FormulaLabel Property	**Range**
GetChartElement Method	**Chart**
GetData Method	**PivotTable**
Has3DEffect Property	**Series**
Has3DShading Property	**ChartGroup**
HasDataTable Property	**Chart**
HasLeaderLines Property	**Series**
Height Property	**Axis**, **LegendEntry**, **LegendKey**
HighlightChangesOnScreen Property	**Workbook**
HighlightChangesOptions Method	**Workbook**
HPageBreaks Property	**Charts**, **Sheets**, **Worksheet**, **Worksheets**
Hyperlinks Property	**Chart**, **Range**, **Worksheet**
IndentLevel Property	**Range**, **Style**
Index Property	**ChartGroup**
InsertIndent Method	**Range**
InsideHeight Property	**PlotArea**
InsideLeft Property	**PlotArea**
InsideTop Property	**PlotArea**
InsideWidth Property	**PlotArea**
InvertIfNegative Property	**Interior**
IsAddin Property	**Workbook**
IsCalculated Property	**PivotField**, **PivotItem**
KeepChangeHistory Property	**Workbook**
LeaderLines Property	**Series**
Left Property	**Axis**, **LegendEntry**, **LegendKey**
LinkedCell Property	**OLEObject**
ListChangesOnNewSheet Property	**Workbook**
ListFillRange Property	**OLEObject**
ListFormulas Method	**PivotTable**
Location Method	**Chart**
MajorUnitScale Property	**Axis**
ManualUpdate Property	**PivotTable**
MarkerSize Property	**LegendKey**, **Point**, **Series**
MemoryUsed Property	**PivotField**
Merge Method	**Range**
MergeArea Property	**Range**
MergeCells Property	**Range**, **Style**
MergeLabels Property	**PivotTable**
MergeWorkbook Method	**Workbook**
MinorUnitScale Property	**Axis**

NewSeries Method	**SeriesCollection**
NullString Property	**PivotTable**
ODBCErrors Property	**Application**
ODBCTimeout Property	**Application**
PageFieldOrder Property	**PivotTable**
PageFieldStyle Property	**PivotTable**
PageFieldWrapCount Property	**PivotTable**
PageRangeCells Property	**PivotTable**
Paste Method	**Floor**, **Walls**
PersonalViewListSettings Property	**Workbook**
PersonalViewPrintSettings Property	**Workbook**
PictureType Property	**Floor**, **LegendKey**, **Walls**
PictureUnit Property	**LegendKey**, **Walls**
PivotCache Method	**PivotTable**
PivotCaches Method	**Workbook**
PivotFormulas Method	**PivotTable**
PivotSelect Method	**PivotTable**
PivotSelection Property	**PivotTable**
PivotTableSelection Property	**Application**
PivotTableWizard Method	**PivotTable**
PlotBy Property	**Chart**
Position Property	**DataLabel**, **DataLabels**
PreserveFormatting Property	**PivotTable**
PrintComments Property	**PageSetup**
ProgId Property	**OLEObject**
ProtectChartObject Property	**ChartObject**
ProtectData Property	**Chart**
ProtectFormatting Property	**Chart**
ProtectGoalSeek Property	**Chart**
ProtectSelection Property	**Chart**
ProtectSharing Method	**Workbook**
PurgeChangeHistoryNow Method	**Workbook**
QueryTable Property	**Range**
QueryTables Property	**Worksheet**
ReadingOrder Property	**AxisTitle**, **ChartTitle**, **DataLabel**, **DataLabels**, **Range**, **Style**, **TickLabels**
RecentFiles Property	**Application**
RecordCount Property	**PivotItem**
Refresh Method	**Chart**
RefreshAll Method	**Workbook**
RejectAllChanges Method	**Workbook**
Reload Method	**Workbook**
RemoveUser Method	**Workbook**

ResetAllPageBreaks Method	**Worksheet**
ResetColors Method	**Workbook**
RollZoom Property	**Application**
SaveWorkspace Method	**Application**
ScrollArea Property	**Worksheet**
SecondaryPlot Property	**Point**
SecondPlotSize Property	**ChartGroup**
SelectionMode Property	**PivotTable**
ServerBased Property	**PivotField**
SetSourceData Method	**Chart**
Shadow Property	**LegendKey**, **Point**, **Series**
ShapeRange Property	**ChartObject**, **ChartObjects**, **OLEObject**, **OLEObjects**
Shapes Property	**Chart**, **Worksheet**
ShowAllItems Property	**PivotField**
ShowChartTipNames Property	**Application**
ShowChartTipValues Property	**Application**
ShowNegativeBubbles Property	**ChartGroup**
ShowWindow Property	**Chart**
ShrinkToFit Property	**Range**, **Style**
SizeRepresents Property	**ChartGroup**
SourceName Property	**OLEObject**, **OLEObjects**
SplitType Property	**ChartGroup**
SplitValue Property	**ChartGroup**
SubtotalHiddenPageItems Property	**PivotTable**
TableStyle Property	**PivotTable**
Tag Property	**PivotTable**
TemplateRemoveExtData Property	**Workbook**
Top Property	**Axis**, **LegendEntry**, **LegendKey**
UILanguage Property	**Application**
UnMerge Method	**Range**
UnprotectSharing Method	**Workbook**
Update Method	**PivotTable**
UserControl Property	**Application**, **Workbook**
VacatedStyle Property	**PivotTable**
Validation Property	**Range**
Value2 Property	**Range**
VBE Property	**Application**
VBProject Property	**Workbook**
View Property	**Window**
VPageBreaks Property	**Charts**, **Sheets**, **Worksheet**, **Worksheets**
Width Property	**Axis**, **LegendEntry**, **LegendKey**
WorksheetFunction Property	**Application**

Hidden Properties and Methods

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscHiddenPMC"}

Hidden properties and methods for visible objects are listed in the following table. Most of the hidden properties and methods have been replaced by new functionality in Microsoft Excel 97. These properties and methods are supported only for backward compatibility; for new code, you should use the replacement functionality provided in Microsoft Excel 97. For more information about the changes to the Microsoft Excel 97 object model, see one of the following topics:

New Objects

New Properties and Methods (by Object)

New Properties and Methods (Alphabetic List)

Hidden Objects

Methods with New Arguments

Changes to Visual Basic in Microsoft Excel for Windows 95

Object	Hidden Property or Method	Replacement
Application	**ActiveDialog**	**ActiveX** controls
Application	**ActiveMenuBar**	**CommandBars**
Chart, **Worksheet**	**Arcs**	**Shapes**
AddIn, **Workbook**	**Author**	**DocumentProperties**
Chart	**AutoFormat**	**ApplyCustomType** method
Chart, **Worksheet**	**Buttons**	**ActiveX** controls
Chart, **Worksheet**	**CheckBoxes**	**ActiveX** controls
Application	**ColorButtons**	no replacement
AddIn, **Workbook**	**Comments**	**DocumentProperties**
Application, **Workbook**	**DialogSheets**	**Custom Forms** and **ActiveX** controls
Worksheet	**DisplayAutomaticPageBreaks**	**DisplayPageBreaks** Property
Application	**DisplayInfoWindow**	no replacement
Chart, **Worksheet**	**DrawingObjects**	**Shapes**
Chart, **Worksheet**	**Drawings**	**Shapes**
Chart, **Worksheet**	**DropDowns**	**ActiveX** controls
Application	**EnableTipWizard**	**Assistant**
ChartObjects, **OLEObjects**	**Group**	**Shapes**
Chart, **Worksheet**	**GroupBoxes**	**Shapes**
Chart, **Worksheet**	**GroupObjects**	**Shapes**
AddIn, **Workbook**	**Keywords**	**DocumentProperties**
Chart, **Worksheet**	**Labels**	**Shapes**
Application	**LargeButtons**	no replacement
Chart, **Worksheet**	**Lines**	**Shapes**
Chart, **Worksheet**	**ListBoxes**	ActiveX controls
Application	**MenuBars**	**CommandBars**
Application, **Workbook**	**Modules**	VBE

ChartObject, **ChartObjects**, **OLEObject**, **OLEObjects**	**OnAction**	**Change** or **SheetChange** event
Application, **Worksheet**	**OnCalculate**	**Calculate** event
Application, **Worksheet**	**OnData**	**Change** or **SheetChange** event
Application, **Chart**, **Worksheet**	**OnDoubleClick**	**BeforeDoubleClick** event
Application, **Worksheet**	**OnEntry**	**Change** event
Workbook	**OnSave**	**BeforeSave** event
Application, **Chart**, **Workbook**, **Worksheet**	**OnSheetActivate**	**Activate** or **SheetActivate** event
Application, **Chart**, **Workbook**, **Worksheet**	**OnSheetDeactivate**	**Deactivate** or **SheetDeactivate** events
Chart, **Worksheet**	**OptionButtons**	**ActiveX** controls
Chart, **Worksheet**	**Ovals**	**ActiveX** controls
Chart, **Worksheet**	**Pictures**	**Shapes**
Chart, **Worksheet**	**Rectangles**	**Shapes**
Application	**ResetTipWizard**	Assistant
Application	**Save**	**SaveWorkspace** Method
Chart, **Worksheet**	**ScrollBars**	**ActiveX** controls
Window	**SetInfoDisplay**	no replacement
Application	**ShortcutMenus**	**CommandBars**
Chart, **Worksheet**	**Spinners**	**ActiveX** controls
AddIn, **Workbook**	**Subject**	**DocumentProperties**
Chart, **ChartGroup**	**SubType**	**ChartType** property
Chart, **Worksheet**	**TextBoxes**	**ActiveX** controls
AddIn, **Workbook**	**Title**	**DocumentProperties**
Application	**Toolbars**	**CommandBars**
Chart, **ChartGroup**	**Type**	**ChartType** property

Methods with New Arguments

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscNewArgsC"}

Methods with arguments that have been added for Microsoft Excel 97 are listed in the following table. For more information about the changes to the Microsoft Excel 97 object model, see one of the following topics:

New Objects

New Properties and Methods (by Object)

New Properties and Methods (Alphabetic List)

Hidden Objects

Hidden Properties and Methods

Changes to Visual Basic in Microsoft Excel for Windows 95

Method	Objects	New arguments
Add	**OLEObjects**	***Left***, ***Top***, ***Width***, ***Height***
ApplyDataLabels	**Chart**, **Series**	***AutoText***, ***HasLeaderLines***
ApplyDataLabels	**Point**	***AutoText***
AutoFilter	**Range**	***VisibleDropDown***
CheckSpelling	**Chart**, **Range**, **Worksheet**	***IgnoreInitialAlefHamza***, ***IgnoreFinalYaa***, ***SpellScript*** (these arguments aren't used in U.S. English Microsoft Excel)
Find	**Range**	***MatchControlCharacters***, ***MatchDiacritics***, ***MatchKashida***, ***MatchAlefHamza*** (these arguments aren't used in U.S. English Microsoft Excel)
Open	**Workbooks**	***AddToMru***
OpenText	**Workbooks**	***TextVisualLayout*** (not used in U.S. English Microsoft Excel)
PivotTableWizard	**Worksheet**	***BackgroundQuery***, ***OptimizeCache***, ***PageFieldOrder***, ***PageFieldWrapCount***, ***ReadData***, ***Connection***
PrintPreview	**Chart**, **Charts**, **Range**, **Sheets**, **Window**, **Workbook**, **Worksheet**, **Worksheets**	***EnableChanges***
RegisteredFunctions	**Application**	***Index1***, ***Index2***
Replace	**Range**	***MatchControlCharacters***, ***MatchDiacritics***, ***MatchKashida***, ***MatchAlefHamza*** (these arguments aren't used in U.S. English Microsoft Excel)
Run	**Application**	Now uses a variable-length argument array

SaveAs	**Chart**, **Workbook**, **Worksheet**	***AddToMru***, ***TextCodepage***, ***TextVisualLayout*** (the last two arguments aren't used in U.S. English Microsoft Excel)
SetDefaultChart	**Application**	***Gallery***
Sort	**Range**	***SortMethod***, ***IgnoreControlCharacters***, ***IgnoreDiacritics***, ***IgnoreKashida*** (the last three arguments aren't used in U.S. English Microsoft Excel)

Changes to Visual Basic in Microsoft Excel for Windows 95

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscChanged95C "} Top

This topic covers changes made to the Visual Basic object model between Microsoft Excel version 5.0 and Microsoft Excel for Windows 95. For more information about changes made between Microsoft Excel for Windows 95 and Microsoft Excel 97, see Changes to the Microsoft Excel 97 Object Model.

Existing properties and methods were changed, and several new properties, methods, and objects were added to Visual Basic in Microsoft Excel for Windows 95. For more information about these changes and additions, select one of the items in the following list, or scroll through this topic to find the information.

Changed Properties And Methods

The following properties and methods have been changed to support new Microsoft Excel features or to enhance existing features.

Topic	Description
GetOpenFileName Method	The new ***multiSelect*** argument allows you to select more than one filename in the Open dialog box.
Names Property	Added to the **Worksheet** object to handle sheet-scoped names.
PasteSpecial Method	For the **Range** object, a new constant for the ***paste*** argument allows you to paste range content and formatting but not border formatting.
Protect Method	The new ***userInterfaceOnly*** argument allows you to specify protection from user changes while still allowing changes from Visual Basic code.
SaveAs Method	New arguments allow you to specify shared-mode access and how change conflicts are resolved.
StatusBar Property	Added to the **ToolbarButton** object.

Autocorrect Support

The following new object, properties, methods have been added to support automatic corrections.

Topic	Description
AddReplacement Method	Adds a row to the array of AutoCorrect replacements returned by the **ReplacementList** method.
AutoCorrect Object	Contains AutoCorrect attributes (capitalization of names of days, correction of two initial capital letters, automatic correction list, and so on).

AutoCorrect Property	Returns an **AutoCorrect** object that represents the AutoCorrect attributes.
CapitalizeNamesOfDays Property	**True** if the first letters of names (and abbreviations) of days are capitalized automatically.
DeleteReplacement Method	Deletes a row from the array of AutoCorrect replacements returned by the **ReplacementList** method.
ReplacementList Method	Returns or sets the entire array or one row of the array of AutoCorrect replacements.
ReplaceText Property	**True** if text from column one of the array of AutoCorrect replacements is automatically replaced with the text from column two.
TwoInitialCapitals Property	**True** if occurrences of two initial capital letters in a word are corrected automatically.

OLE Document Properties Support

The following new properties, methods, and objects have been added to support OLE document properties. Document properties allow Microsoft Excel to expose a standard set of built-in document properties and additional custom document properties added by the user.

Topic	Description
BuiltinDocumentProperties Property	Returns a **DocumentProperties** collection object that contains the built-in document properties for the object.
CustomDocumentProperties Property	Returns a **DocumentProperties** collection object that contains the custom document properties for the object.
DocumentProperties Object	Represents the collection of document properties (either built-in or custom).
DocumentProperty Object	Represents a single document property.
LinkedToContent Property	**True** if the value of a custom property is linked to the contents of the document it's contained in; **False** if the property is a static value.
LinkSource Property	Specifies the source of a custom document property that's linked to document contents.

Autocomplete Support

The following method and property have been added to support the AutoComplete feature.

Topic	Description
AutoComplete Method	Returns a completed string from the list, or returns an empty string if there was no completion or if more than one entry in the list matches.
EnableAutoComplete Property	**True** if the AutoComplete feature is enabled.

Protection Control

The following properties have been added to allow you to control the user's ability to change toolbars, filtered lists, PivotTables, and outlining.

Topic	Description
EnableAutoFilter Property	**True** if AutoFilter arrows are enabled when user-interface-only

	protection is turned on.
EnableOutlining Property	**True** if outlining symbols (show detail or hide detail) are enabled when user-interface-only protection is turned on.
EnablePivotTable Property	**True** if PivotTable controls and actions are enabled when user-interface-only protection is turned on

Appearance Control

The following properties and methods have been added to control the appearance of the mouse pointer, turn animated insertion and deletion on and off, create a password-entry edit box, and set the background graphic for a worksheet or chart.

Topic	Description
Cursor Property	Sets the appearance of the mouse pointer in Microsoft Excel.
EnableAnimations Property	**True** if animated insertion and deletion is enabled.
SetBackgroundPicture Method	Sets the background graphic for a worksheet or chart.

Miscellaneous

Topic	Description
AutoLoad Property	**True** if the OLE object is automatically loaded when the workbook that contains the object is opened.
CopyFromRecordset Method	Copies the contents of a DAO **Recordset** object into cells on a worksheet, beginning at the first cell of the specified range.
ListHeaderRows Property	Returns the number of header rows for the specified range.
MoveAfterReturnDirection Property	Controls the direction in which the active cell is moved after the user presses ENTER.
NetworkTemplatesPath Property	Returns the network path where templates are stored. If the network path doesn't exist, the property returns an empty string.
Post Method	Posts the specified workbook to a Microsoft Exchange public folder or a Lotus Notes database.
ProtectionMode Property	**True** if user-interface-only protection is turned on.
RangeSelection Property	Returns a **Range** object that represents the selected cells on the worksheet in the specified window even if a graphic object is active or selected on the worksheet.
RefersToRange Property	Returns the **Range** object referred to by a **Name** object.
TemplatesPath Property	Returns the local path where templates are stored.

RunAutoMacros Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRunAutoMacrosC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRunAutoMacrosX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRunAutoMacrosA "}

Runs the Auto_Open, Auto_Close, Auto_Activate, or Auto_Deactivate macro attached to the workbook. This method is included for backward compatibility. For new Visual Basic code, you should use the Open, Close, Activate and Deactivate events instead of these macros.

Syntax

expression.**RunAutoMacros**(***Which***)

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

Which Required. The macros to run. Can be one of the following **XlRunAutoMacro** constants:

Constant	Description
xlAutoOpen	Auto_Open macros
xlAutoClose	Auto_Close macros
xlAutoActivate	Auto_Activate macros
xlAutoDeactivate	Auto_Deactivate macros

RunAutoMacros Method Example

This example opens the workbook Analysis.xls and then runs its Auto_Open macro.

```
Workbooks.Open "ANALYSIS.XLS"
ActiveWorkbook.RunAutoMacros xlAutoOpen
```

This example runs the Auto_Close macro for the active workbook and then closes the workbook.

```
With ActiveWorkbook
   .RunAutoMacros xlAutoClose
   .Close
End With
```

Activate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtActivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtActivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtActivateA"}

Occurs when a workbook, worksheet, chart sheet, or embedded chart is activated.

Syntax

Private Sub *object*_**Activate()**

object **Chart**, **Workbook**, or **Worksheet**. For information about using events with the **Chart** object, see Using Events with the Chart Object.

Remarks

When you switch between two windows showing the same workbook, the WindowActivate event occurs, but the Activate event for the workbook doesn't occur.

This event doesn't occur when you create a new window.

Activate Event Example

This example sorts the range A1:A10 when the worksheet is activated.

```
Private Sub Worksheet_Activate()
   Range("a1:a10").Sort Key1:=Range("a1"), Order:=xlAscending
End Sub
```

AddinInstall Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtAddinInstallC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtAddinInstallX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtAddinInstallA"}

Occurs when the workbook is installed as an add-in

Syntax

Private Sub Workbook_AddinInstall()

AddinInstall Event Example

This example adds a control to the standard toolbar when the workbook is installed as an add-in.

```
Private Sub Workbook_AddinInstall()
   With Application.Commandbars("Standard").Controls.Add
      .Caption = "The AddIn's menu item"
      .OnAction = "'ThisAddin.xls'!Amacro"
   End With End Sub
End Sub
```

AddinUninstall Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtAddinUninstallC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtAddinUninstallX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtAddinUninstallA"}

Occurs when the workbook is uninstalled as an add-in.

Syntax

Private Sub Workbook_AddinUninstall()

Remarks

The add-in doesn't automatically close when it's uninstalled.

AddinUninstall Event Example

This example minimizes Microsoft Excel when the workbook is uninstalled as an add-in.

```
Private Sub Workbook_AddinUninstall()
   Application.WindowState = xlMinimized
End Sub
```

AfterRefresh Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtAfterRefreshC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtAfterRefreshX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtAfterRefreshA"}

Occurs after a query is completed or canceled.

Syntax

Private Sub QueryTable_AfterRefresh(*Success* **As Boolean)**

Success **True** if the query was completed successfully.

AfterRefresh Event Example

This example uses the `Success` argument to determine which section of code to run.

```
Private Sub QueryTable_AfterRefresh(Success As Boolean)
   If Success
      ' Query completed successfully
   Else
      ' Query failed or was cancelled
   End If
End Sub
```

BeforeClose Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtBeforeCloseC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtBeforeCloseX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtBeforeCloseA"}

Occurs before the workbook closes. If the workbook has been changed, this event occurs before the user is asked to save changes.

Syntax

Private Sub Workbook_BeforeClose(*Cancel* **As Boolean)**

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the close operation stops and the workbook is left open.

BeforeClose Event Example

This example always saves the workbook if it's been changed.

```
Private Sub Workbook_BeforeClose(Cancel as Boolean)
   If Me.Saved = False Then Me.Save
End Sub
```

BeforeDoubleClick Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtBeforeDoubleClickC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtBeforeDoubleClickX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtBeforeDoubleClickA"}

Occurs when an embedded chart or worksheet is double-clicked, before the default double-click action.

Syntax 1

Private Sub Worksheet_BeforeDoubleClick(ByVal *Target* **As Range,** *Cancel* **As Boolean)**

Syntax 2

Private Sub *object*_**BeforeDoubleClick(ByVal** *ElementID* **As Long, ByVal** *Arg1* **As Long, ByVal** *Arg2* **As Long,** *Cancel* **As Boolean)**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

Target The cell nearest to the mouse pointer when the double-click occurs.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the default double-click action isn't performed when the procedure is finished.

ElementID The double-clicked object The meaning of *Arg1* and *Arg2* depends on the *ElementID* value, as shown in the following table.

ElementID	Arg1	Arg2
xlChartArea	None	None
xlChartTitle	None	None
xlPlotArea	None	None
xlLegend	None	None
xlFloor	None	None
xlWalls	None	None
xlCorners	None	None
xlDataTable	None	None
xlSeries	SeriesIndex	PointIndex
xlDataLabel	SeriesIndex	PointIndex
xlTrendline	SeriesIndex	TrendLineIndex
xlErrorBars	SeriesIndex	None
xlXErrorBars	SeriesIndex	None
xlYErrorBars	SeriesIndex	None
xlLegendEntry	SeriesIndex	None
xlLegendKey	SeriesIndex	None
xlAxis	AxisIndex	AxisType
xlMajorGridlines	AxisIndex	AxisType
xlMinorGridlines	AxisIndex	AxisType
xlAxisTitle	AxisIndex	AxisType
xlUpBars	GroupIndex	None
xlDownBars	GroupIndex	None
xlSeriesLines	GroupIndex	None
xlHiLoLines	GroupIndex	None
xlDropLines	GroupIndex	None
xlRadarAxisLabels	GroupIndex	None

xlShape	ShapeIndex	None
xlNothing	None	None

The following table describes the meaning of the arguments.

Argument	Description
SeriesIndex	Specifies the offset within the **Series** collection for a specific series.
PointIndex	Specifies the offset within the **Points** collection for a specific point within a series. The value –1 indicates that all data points are selected.
TrendlineIndex	Specifies the offset within the **Trendlines** collection for a specific trendline within a series.
AxisIndex	Specifies whether the axis is primary (0) or secondary (1).
AxisType	Specifies the axis type: category (0), value (1), or series (2).
GroupIndex	Specifies the offset within the **ChartGroups** collection for a specific chart group.
ShapeIndex	Specifies the offset within the **Shapes** collection for a specific shape.

Remarks

The **DoubleClick** method doesn't cause this event to occur.

This event doesn't occur when the user double-clicks the border of a cell.

BeforeDoubleClick Event Example

This example overrides the default double-click behavior for the chart floor.

```
Private Sub Chart_BeforeDoubleClick(ByVal ElementID As Long, _
      ByVal Arg1 As Long, ByVal Arg2 As Long, Cancel As Boolean)
   If ElementID = xlFloor Then
      Cancel = True
      MsgBox "Chart formatting for this item is restricted."
   End If
End Sub
```

BeforePrint Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtBeforePrintC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtBeforePrintX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtBeforePrintA"}

Occurs before the workbook (or anything in it) is printed.

Syntax

Private Sub Workbook_BeforePrint(*Cancel* **As Boolean)**

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the workbook isn't printed when the procedure is finished.

BeforePrint Event Example

This example recalculates all worksheets in the active workbook before printing anything.

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
   For Each wk in Worksheets
      wk.Calculate
   Next
End Sub
```

BeforeRefresh Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtBeforeRefreshC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtBeforeRefreshX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtBeforeRefreshA"}

Occurs before any refreshes of the query table. This includes refreshes resulting from calling the **Refresh** method, from the user's actions in the product, and from opening the workbook containing the query table.

Syntax

Private Sub QueryTable_BeforeRefresh(*Cancel* **As Boolean)**

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the refresh doesn't occur when the procedure is finished.

BeforeRefresh Event Example

This example runs before the query table is refreshed.

```
Private Sub QueryTable_BeforeRefresh(Cancel As Boolean)
    a = MsgBox("Refresh Now?", vbYesNoCancel)
    If a = vbNo Then Cancel = True
    MsgBox Cancel
End Sub
```

BeforeRightClick Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtBeforeRightClickC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtBeforeRightClickX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtBeforeRightClickA"}

Occurs when an embedded chart or worksheet is right-clicked, before the default right-click action.

Syntax 1

Private Sub *object*_**BeforeRightClick(***Cancel* **As Boolean)**

Syntax 2

Private Sub Worksheet_BeforeRightClick(ByVal *Target* **As Range,** *Cancel* **As Boolean)**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

Target The cell nearest to the mouse pointer when the right-click occurs.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the default right-click action doesn't occur when the procedure is finished.

Remarks

Like other worksheet events, this event doesn't occur if you right-click while the pointer is on a shape or a command bar (a toolbar or menu bar).

BeforeRightClick Event Example

This example adds a new menu item to the shortcut menu for cells B1:B10.

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, _
        Cancel As Boolean)
    For Each icbc In Application.CommandBars("cell").Controls
        If icbc.Tag = "brccm" Then icbc.Delete
    Next icbc
    If Not Application.Intersect(Target, Range("b1:b10")) _
        Is Nothing Then
            With Application.CommandBars("cell").Controls _
                .Add(Type:=msoControlButton, before:=6, _
                temporary:=True)
           .Caption = "New Context Menu Item"
           .OnAction = "MyMacro"
           .Tag = "brccm"
        End With
    End If
End Sub
```

BeforeSave Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtBeforeSaveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtBeforeSaveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtBeforeSaveA"}

Occurs before the workbook is saved.

Syntax

Private Sub Workbook_BeforeSave(ByVal *SaveAsUi* **As Boolean,** *Cancel* **As Boolean)**

SaveAsUi **True** if the **Save As** dialog box will be displayed.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the workbook isn't saved when the procedure is finished.

BeforeSave Event Example

This example prompts the user for a yes or no response before saving the workbook.

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, _
      Cancel as Boolean)
   a = MsgBox("Do you really want to save the workbook?", vbYesNo)
   If a = vbNo Then Cancel = True
End Sub
```

Calculate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtCalculateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtCalculateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtCalculateA"}

Chart object: Occurs after the chart plots new or changed data.

Worksheet object: Occurs after the worksheet is recalculated.

Syntax

Private Sub *object*_**Calculate()**

object **Chart** or **Worksheet**. For information about using events with the **Chart** object, see Using Events with the Chart Object.

Calculate Event Example

This example adjusts the size of columns A through F whenever the worksheet is recalculated.

```
Private Sub Worksheet_Calculate()
   Columns("A:F").AutoFit
End Sub
```

Change Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtChangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtChangeA"}

Occurs when cells on the worksheet are changed by the user or by an external link.

Syntax

Private Sub Worksheet_Change(ByVal *Target* **As Range)**

Target The changed range. Can be more than one cell.

Remarks

This event doesn't occur when cells change during a recalculation. Use the Calculate event to trap a sheet recalculation.

Deleting cells doesn't trigger this event.

Change Event Example

This example changes the color of changed cells to blue.

```
Private Sub Worksheet_Change(ByVal Target as Range)
   Target.Font.ColorIndex = 5
End Sub
```

Deactivate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtDeactivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtDeactivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtDeactivateA"}

Occurs when the chart, worksheet, or workbook is deactivated.

Syntax

Private Sub *object*_**Deactivate()**

object **Chart**, **Workbook**, or **Worksheet**. For information about using events with the **Chart** object, see Using Events with the Chart Object.

Deactivate Event Example

This example arranges all open windows when the workbook is deactivated.

```
Private Sub Workbook_Deactivate()
   Application.Windows.Arrange xlArrangeStyleTiled
End Sub
```

DragOver Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtDragOverC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtDragOverX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtDragOverA"}

Occurs when a range of cells is dragged over a chart.

Syntax

Private Sub *object*_**DragOver()**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

DragOver Event Example

This example displays the address of a range of cells dragged over a chart.

```
Private Sub Chart_DragOver()
   MsgBox Selection.Address
End Sub
```

DragPlot Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtDragPlotC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtDragPlotX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtDragPlotA"}

Occurs when a range of cells is dragged and dropped on a chart.

Syntax

Private Sub *object*_**DragPlot()**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

DragPlot Event Example

This example changes the chart type when a range of cells is dragged and dropped on a chart.

```
Private Sub Chart_DragPlot()
   Me.ChartType = xlLine
End Sub
```

MouseDown Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtMouseDownC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtMouseDownX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtMouseDownA"}

Occurs when a mouse button is pressed while the pointer is over a chart.

Syntax

Private Sub *object*_**MouseDown(ByVal** *Button* **As Long, ByVal** *Shift* **As Long, ByVal** *X* **As Long, ByVal** *Y* **As Long)**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

Button The mouse button that was pressed. Can be one of the following **XlMouseButton** constants: **xlNoButton**, **xlPrimaryButton**, **xlSecondaryButton**, or **xlMiddleButton**.

Shift The state of the SHIFT, CTRL, and ALT keys when the event occurred. Can be one of or a sum of the following values.

Value	Meaning
0 (zero)	No keys
1	SHIFT key
2	CTRL key
4	ALT key

X The X coordinate of the mouse pointer in chart object client coordinates.

Y The Y coordinate of the mouse pointer in chart object client coordinates.

MouseDown Event Example

This example runs when a mouse button is pressed while the pointer is over a chart.

```
Private Sub Chart_MouseDown(ByVal Button As Long, _
      ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
   MsgBox "Button = " & Button & chr$(13) & _
      "Shift = " & Shift & chr$(13) & _
      "X = " & X & " Y = " & Y
End Sub
```

MouseMove Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtMouseMoveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtMouseMoveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtMouseMoveA"}

Occurs when the position of the mouse pointer changes over a chart.

Syntax

Private Sub *object*_**MouseMove(ByVal** *X* **As Long, ByVal** *Y* **As Long)**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

X The X coordinate of the mouse pointer in chart object client coordinates.

Y The Y coordinate of the mouse pointer in chart object client coordinates.

MouseMove Event Example

This example runs when the position of the mouse pointer changes over a chart.

```
Private Sub Chart_MouseMove(ByVal X As Long, ByVal Y As Long)
   MsgBox "X = " & X & " Y = " & Y
End Sub
```

MouseUp Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtMouseUpC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtMouseUpX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtMouseUpA"}

Occurs when a mouse button is released while the pointer is over a chart.

Syntax

Private Sub *object*_**MouseUp(ByVal** *Button* **As Long, ByVal** *Shift* **As Long, ByVal** *X* **As Long, ByVal** *Y* **As Long)**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

Button The mouse button that was released. Can be one of the following **XlMouseButton** constants: **xlNoButton**, **xlPrimaryButton**, **xlSecondaryButton**, or **xlMiddleButton**.

Shift The state of the SHIFT, CTRL, and ALT keys when the event occurred. Can be one of or a sum of the following values.

Value	Meaning
0 (zero)	No keys
1	SHIFT key
2	CTRL key
4	ALT key

X The X coordinate of the mouse pointer in chart object client coordinates.

Y The Y coordinate of the mouse pointer in chart object client coordinates.

MouseUp Event Example

This example runs when a mouse button is released over a chart.

```
Private Sub Chart_MouseUp(ByVal Button As Long, _
      ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
   MsgBox "Button = " & Button & chr$(13) & _
      "Shift = " & Shift & chr$(13) & _
      "X = " & X & " Y = " & Y
End Sub
```

NewSheet Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtNewSheetC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtNewSheetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtNewSheetA"}

Occurs when a new sheet is created in the workbook.

Syntax

Private Sub Workbook_NewSheet(ByVal *Sh* **As Object)**

Sh The new sheet. Can be a **Worksheet** or **Chart** object.

NewSheet Event Example

This example moves new sheets to the end of the workbook.

```
Private Sub Workbook_NewSheet(ByVal Sh as Object)
   Sh.Move After:= Sheets(Sheets.Count)
End Sub
```

NewWorkbook Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtNewWorkbookC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtNewWorkbookX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtNewWorkbookA"}

Occurs when a new workbook is created.

Syntax

Private Sub *object*_**NewWorkbook(ByVal** *Wb* **As Workbook)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The new workbook.

NewWorkbook Event Example

This example arranges open windows when a new workbook is created.

```
Private Sub App_NewWorkbook(ByVal Wb As Workbook)
   Application.Windows.Arrange xlArrangeStyleTiled
End Sub
```

SeriesChange Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSeriesChangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSeriesChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSeriesChangeA"}

Occurs when the user changes the value of a chart data point.

Syntax

Private Sub *object*_**SeriesChange(ByVal** *SeriesIndex* **As Long, ByVal** *PointIndex* **As Long)**

object An object of type **Chart** declared with events in a class module. For more information, see Using Events with the Chart Object.

SeriesIndex The offset within the **Series** collection for the changed series.

PointIndex The offset within the **Points** collection for the changed point.

SeriesChange Event Example

This example changes the point's border color when the user changes the point value.

```
Private Sub Chart_SeriesChange(ByVal SeriesIndex As Long, _
      ByVal PointIndex As Long)
    Set p = Me.SeriesCollection(SeriesIndex).Points(PointIndex)
    p.Border.ColorIndex = 3
End Sub
```

SheetActivate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSheetActivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSheetActivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSheetActivateA"}

Occurs when any sheet is activated.

Syntax

Private Sub *object*_**SheetActivate(ByVal** *Sh* **As Object)**

object **Application** or **Workbook**.

Sh The activated sheet. Can be a **Chart** or **Worksheet** object.

SheetActivate Event Example

This example displays the name of each activated sheet.

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
   MsgBox Sh.Name
End Sub
```

SheetBeforeDoubleClick Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSheetBeforeDoubleClickC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSheetBeforeDoubleClickX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSheetBeforeDoubleClickA"}

Occurs when any worksheet is double-clicked, before the default double-click action.

Syntax

Private Sub *object*_**SheetBeforeDoubleClick(ByVal** *Sh* **As Object, ByVal** *Target* **As Range, ByVal** *Cancel* **As Boolean)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Sh A **Worksheet** object that represents the sheet.

Target The cell nearest to the mouse pointer when the double-click occurred.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the default double-click action isn't performed when the procedure is finished.

Remarks

This event doesn't occur on chart sheets.

SheetBeforeDoubleClick Event Example

This example disables the default double-click action.

```
Private Sub Workbook_SheetBeforeDoubleClick(ByVal Sh As Object, _
      ByVal Target As Range, ByVal Cancel As Boolean)
   Cancel = True
End Sub
```

SheetBeforeRightClick Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSheetBeforeRightClickC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSheetBeforeRightClickX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSheetBeforeRightClickA"}

Occurs when any worksheet is right-clicked, before the default right-click action.

Syntax

Private Sub *object*_**SheetBeforeRightClick(ByVal** *Sh* **As Object, ByVal** *Target* **As Range, ByVal** *Cancel* **As Boolean)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Sh A **Worksheet** object that represents the sheet.

Target The cell nearest to the mouse pointer when the right-click occurred.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the default right-click action isn't performed when the procedure is finished.

Remarks

This event doesn't occur on chart sheets.

SheetBeforeRightClick Event Example

This example disables the default right-click action. For another example, see the BeforeRightClick event example.

```
Private Sub Workbook_SheetBeforeRightClick(ByVal Sh As Object, _
      ByVal Target As Range, ByVal Cancel As Boolean)
   Cancel = True
End Sub
```

SheetCalculate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSheetCalculateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSheetCalculateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSheetCalculateA"}

Occurs after any worksheet is recalculated or after any changed data is plotted on a chart.

Syntax

Private Sub *object*_**SheetCalculate(ByVal** *Sh* **As Object)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Sh The sheet. Can be a **Chart** or **Worksheet** object.

SheetCalculate Event Example

This example sorts the range A1:A100 on worksheet one when any sheet in the workbook is calculated.

```
Private Sub Workbook_SheetCalculate(ByVal Sh As Object)
   With Worksheets(1)
      .Range("a1:a100").Sort Key1:=.Range("a1")
   End With
End Sub
```

SheetChange Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSheetChangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSheetChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSheetChangeA"}

Occurs when cells in any worksheet are changed by the user or by an external link.

Syntax

Private Sub *object*_**SheetChange(ByVal** *Sh* **As Object, ByVal** *Source* **As Range)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Sh A **Worksheet** object that represents the sheet.

Source The changed range.

Remarks

This event doesn't occur on chart sheets.

SheetChange Event Example

This example runs when any worksheet is changed.

```
Private Sub Workbook_SheetChange(ByVal Sh As Object, _
      ByVal Source As Range)
   ' runs when a sheet is changed
End Sub
```

SheetDeactivate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSheetDeactivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSheetDeactivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSheetDeactivateA"}

Occurs when any sheet is deactivated.

Syntax

Private Sub *object*_**SheetDeactivate(ByVal** *Sh* **As Object)**

object **Application** or **Workbook**.

Sh The sheet. Can be a **Chart** or **Worksheet** object.

SheetDeactivate Event Example

This example displays the name of each deactivated sheet.

```
Private Sub Workbook_SheetDeactivate(ByVal Sh As Object)
   MsgBox Sh.Name
End Sub
```

WorkbookActivate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookActivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookActivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookActivateA"}

Occurs when any workbook is activated.

Syntax

Private Sub *app*_**WorkbookActivate(ByVal** *Wb* **As Workbook)**

app An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The activated workbook.

WorkbookActivate Event Example

This example arranges open windows when a workbook is activated.

```
Private Sub App_WorkbookActivate(ByVal Wb As Workbook)
   Application.Windows.Arrange xlArrangeStyleTiled
End Sub
```

WorkbookAddinInstall Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookAddinInstallC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookAddinInstallX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookAddinInstallA"}

Occurs when a workbook is installed as an add-in.

Syntax

Private Sub *object*_**WorkbookAddinInstall(ByVal** *Wb* **As Workbook)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The installed workbook.

WorkbookAddinInstall Event Example

This example maximizes the Microsoft Excel window when a workbook is installed as an add-in.

```
Private Sub App_WorkbookAddinInstall(ByVal Wb As Workbook)
   Application.WindowState = xlMaximized
End Sub
```

WorkbookAddinUninstall Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookAddinUninstallC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookAddinUninstallX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookAddinUninstallA"}

Occurs when any add-in workbook is uninstalled.

Syntax

Private Sub *object*_**WorkbookAddinUninstall(ByVal** *Wb* **As Workbook)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The uninstalled workbook.

WorkbookAddinUninstall Event Example

This example minimizes the Microsoft Excel window when a workbook is installed as an add-in.

```
Private Sub App_WorkbookAddinUninstall(ByVal Wb As Workbook)
   Application.WindowState = xlMinimized
End Sub
```

WorkbookBeforeClose Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookBeforeCloseC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookBeforeCloseX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookBeforeCloseA"}

Occurs immediately before any open workbook closes.

Syntax

Private Sub *object*_**WorkbookBeforeClose(ByVal** *Wb* **As Workbook, ByVal** *Cancel* **As Boolean)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The workbook that's being closed.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the workbook doesn't close when the procedure is finished.

WorkbookBeforeClose Event Example

This example prompts the user for a yes or no response before closing any workbook.

```
Private Sub App_WorkbookBeforeClose(ByVal Wb as Workbook, _
      Cancel as Boolean)
   a = MsgBox("Do you really want to close the workbook?", vbYesNo)
   If a = vbNo Then Cancel = True
End Sub
```

WorkbookBeforePrint Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookBeforePrintC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookBeforePrintX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookBeforePrintA"}

Occurs before any open workbook is printed.

Syntax

Private Sub *object*_**WorkbookBeforePrint(ByVal** *Wb* **As Workbook, ByVal Cancel As Boolean)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The workbook.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the workbook isn't printed when the procedure is finished.

WorkbookBeforePrint Event Example

This example recalculates all worksheets in the workbook before printing anything.

```
Private Sub App_WorkbookBeforePrint(ByVal Wb As Workbook, _
      Cancel As Boolean)
   For Each wk in Wb.Worksheets
      wk.Calculate
   Next
End Sub
```

WorkbookBeforeSave Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookBeforeSaveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookBeforeSaveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookBeforeSaveA"}

Occurs before any open workbook is saved.

Syntax

Private Sub *object*_**WorkbookBeforeSave(ByVal** *Wb* **As Workbook, ByVal** *SaveAsUi* **As Boolean, ByVal** *Cancel* **As Boolean)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The workbook.

SaveAsUi **True** if the **Save As** dialog box will be displayed.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the workbook isn't saved when the procedure is finished.

WorkbookBeforeSave Event Example

This example prompts the user for a yes or no response before saving any workbook.

```
Private Sub App_WorkbookBeforeSave(ByVal Wb As Workbook, _
      ByVal SaveAsUI As Boolean, Cancel as Boolean)
   a = MsgBox("Do you really want to save the workbook?", vbYesNo)
   If a = vbNo Then Cancel = True
End Sub
```

WorkbookDeactivate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookDeactivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookDeactivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookDeactivateA"}

Occurs when any open workbook is deactivated.

Syntax

Private Sub *object*_**WorkbookDeactivate(ByVal** *Wb* **As Workbook)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The workbook.

WorkbookDeactivate Event Example

This example arranges all open windows when a workbook is deactivated.

```
Private Sub App_WorkbookDeactivate(ByVal Wb As Workbook)
   Application.Windows.Arrange xlArrangeStyleTiled
End Sub
```

WorkbookNewSheet Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookNewSheetC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookNewSheetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookNewSheetA"}

Occurs when a new sheet is created in any open worbook.

Syntax

Private Sub *object*_**WorkbookNewSheet(ByVal** *Wb* **As Workbook, ByVal** *Sh* **As Object)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The workbook.

Sh The new sheet.

WorkbookNewSheet Event Example

This example moves the new sheet to the end of the workbook.

```
Private Sub App_WorkbookNewSheet(ByVal Wb As Workbook, _
      ByVal Sh As Object)
   Sh.Move After:=Wb.Sheets(Wb.Sheets.Count)
End Sub
```

WorkbookOpen Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWorkbookOpenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWorkbookOpenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWorkbookOpenA"}

Occurs when a workbook is opened.

Syntax

Private Sub *object*_**WorkbookOpen(ByVal** *Wb* **As Workbook)**

object An object of type **Application** declared with events in a class module. For more information, see Using Events with the Application Object.

Wb The workbook.

WorkbookOpen Event Example

This example arranges all open windows when a workbook is opened.

```
Private Sub  App_WorkbookOpen(ByVal Wb As Workbook)
   Application.Windows.Arrange xlArrangeStyleTiled
End Sub
```

Using Events with the Application Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowUsingAppEventsC"}

Before you can use events with the **Application** object, you must create a new class module and declare an object of type **Application** with events. For example, assume that a new class module is created and called EventClassModule. The new class module contains the following code.

```
Public WithEvents App As Application
```

After the new object has been declared with events, it appears in the **Object** drop-down list box in the class module, and you can write event procedures for the new object. (When you select the new object in the **Object** box, the valid events for that object are listed in the **Procedure** drop-down list box.)

Before the procedures will run, however, you must connect the declared object in the class module with the **Application** object. You can do this with the following code from any module.

```
Dim X As New EventClassModule

Sub InitializeApp()
   Set X.App = Application
End Sub
```

After you run the InitializeApp procedure, the App object in the class module points to the Microsoft Excel **Application** object, and the event procedures in the class module will run when the events occur.

GotFocus Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtGotFocusC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtGotFocusX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtGotFocusA"}

Occurs when an ActiveX control gets input focus.

Syntax

Private Sub *object*_**GotFocus()**

object The name of an ActiveX control.

GotFocus Event Example

This example runs when ListBox1 gets the focus.

```
Private Sub ListBox1_GotFocus()
    ' runs when list box gets the focus
End Sub
```

LostFocus Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtLostFocusC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtLostFocusX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtLostFocusA"}

Occurs when an ActiveX control loses input focus.

Syntax

Private Sub *object*_**LostFocus()**

object The name of an ActiveX control.

LostFocus Event Example

This example runs when ListBox1 loses the focus.

```
Private Sub ListBox1_LostFocus()
    ' runs when list box loses the focus
End Sub
```

Open Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtOpenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtOpenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtOpenA"}

Occurs when the workbook is opened.

Syntax

Private Sub Workbook_Open()

Open Event Example

This example maximizes Microsoft Excel whenever the workbook is opened.

```
Private Sub Workbook_Open()
   Application.WindowState = xlMaximized
End Sub
```

Resize Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtResizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtResizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtResizeA"}

Occurs when the chart is resized.

Syntax

Private Sub *object*_**Resize()**

object **Chart** or an object of type **Chart** declared with events in a class module. For more information, see Using Events with Embedded Charts.

Resize Event Example

This example keeps the upper-left corner of the chart at the same location when the chart is resized.

```
Private Sub myChartClass_Resize()
    With ActiveChart.Parent
        .Left = 100
        .Top = 150
    End With
End Sub
```

Select Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSelectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSelectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSelectA"}

Occurs when a chart element is selected.

Syntax

Private Sub *object*_**Select(ByVal** *ElementID* **As Long, ByVal** *Arg1* **As Long, ByVal** *Arg2* **As Long)**

object **Chart** or an object of type **Chart** declared with events in a class module. For more information, see Using Events with Embedded Charts.

ElementID, *Arg1*, *Arg2* The selected chart element. For more information about these arguments, see the BeforeDoubleClick event.

Select Event Example

This example displays a message box if the user selects the chart title.

```
Private Sub Chart_Select(ByVal ElementID As Long, _
      ByVal Arg1 As Long, ByVal Arg2 As Long)
   If ElementId = xlChartTitle Then
      MsgBox "please don't change the chart title"
   End If
End Sub
```

SelectionChange Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSelectionChangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSelectionChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSelectionChangeA"}

Occurs when the selection changes on a worksheet.

Syntax

Private Sub Worksheet_SelectionChange(ByVal *Target* **As Excel.Range)**

Target The new selected range.

SelectionChange Event Example

This example scrolls through the workbook window until the selection is in the upper-left corner of the window.

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
   With ActiveWindow
      .ScrollRow = Target.Row
      .ScrollColumn = Target.Column
End Sub
```

SheetSelectionChange Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtSheetSelectionChangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtSheetSelectionChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtSheetSelectionChangeA"}

Occurs when the selection changes on any worksheet (doesn't occur if the selection is on a chart sheet).

Syntax

Private Sub *object*_**SheetSelectionChange(ByVal** *Sh* **As Object, ByVal** *Target* **As Excel.Range)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Sh The worksheet that contains the new selection.

Target The new selected range.

SheetSelectionChange Event Example

This example displays the sheet name and address of the selected range in the status bar.

```
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, _
      ByVal Target As Excel.Range)
   Application.StatusBar = Sh.Name & ":" & Target.Address
End Sub
```

WindowActivate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWindowActivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWindowActivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWindowActivateA"}

Occurs when any workbook window is activated.

Syntax

Private Sub *object*_**WindowActivate(ByVal** *Wb* **As Excel.Workbook, ByVal** *Wn* **As Excel.Window)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Wb Used only with the **Application** object. The workbook displayed in the activated window.

Wn The activated window.

WindowActivate Event Example

This example maximizes any workbook window when it's activated.

```
Private Sub Workbook_WindowActivate(ByVal Wn As Excel.Window)
   Wn.WindowState = xlMaximized
End Sub
```

WindowDeactivate Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWindowDeactivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWindowDeactivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWindowDeactivateA"}

Occurs when any workbook window is deactivated.

Syntax

Private Sub *object*_**WindowDeactivate(ByVal** *Wb* **As Excel.Workbook, ByVal** *Wn* **As Excel.Window)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Wb Used only with the **Application** object. The workbook displayed in the deactivated window.

Wn The deactivated window.

WindowDeactivate Event Example

This example minimizes any workbook window when it's deactivated.

```
Private Sub Workbook_WindowDeactivate(ByVal Wn As Excel.Window)
   Wn.WindowState = xlMinimized
End Sub
```

WindowResize Event

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlevtWindowResizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlevtWindowResizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlevtWindowResizeA"}

Occurs when any workbook window is resized.

Syntax

Private Sub *object*_**WindowResize(ByVal** *Wb* **As Excel.Workbook, ByVal** *Wn* **As Excel.Window)**

object **Application** or **Workbook**. For more information about using events with the **Application** object, see Using Events with the Application Object.

Wb Used only with the **Application** object. The workbook displayed in the resized window.

Wn The resized window.

WindowResize Event Example

This example runs when any workbook window is resized.

```
Private Sub Workbook_WindowResize(ByVal Wn As Excel.Window)
   Application.StatusBar = Wn.Caption & " resized"
End Sub
```

Using Events with Embedded Charts

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowUsingChartEventsC"}

Events are enabled for chart sheets by default. Before you can use events with a **Chart** object that represents an embedded chart, you must create a new class module and declare an object of type **Chart** with events. For example, assume that a new class module is created and named "EventClassModule." The new class module contains the following code.

```
Public WithEvents myChartClass As Chart
```

After the new object has been declared with events, it appears in the **Object** drop-down list box in the class module, and you can write event procedures for this object. (When you select the new object in the **Object** box, the valid events for that object are listed in the **Procedure** drop-down list box.)

Before your procedures will run, however, you must connect the declared object in the class module with the embedded chart. You can do this by using the following code from any module.

```
Dim myClassModule As New EventClassModule

Sub InitializeChart()
   Set myClassModule.myChartClass = _
      Worksheets(1).ChartObjects(1).Chart
End Sub
```

After you run the InitializeChart procedure, the `myChartClass` object in the class module points to embedded chart one on worksheet one, and the event procedures in the class module will run when the events occur.

Looping through a range of cells

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowHowToLoopThroughARangeOfCellsC"}

When using Visual Basic, you often need to run the same block of statements on each cell in a range of cells. To do this, you combine a looping statement and one or more methods to identify each cell, one at a time, and run the operation.

One way to loop through a range is to use the **For...Next** loop with the **Cells** property. Using the **Cells** property, you can substitute the loop counter (or other variables or expressions) for the cell index numbers. In the following example, the variable `counter` is substituted for the row index. The procedure loops through the range C1:C20, setting to 0 (zero) any number whose absolute value is less than 0.01.

```
Sub RoundToZero1()
   For counter = 1 To 20
      Set curCell = Worksheets("Sheet1").Cells(counter, 3)
      If Abs(curCell.Value) < 0.01 Then curCell.Value = 0
   Next counter
End Sub
```

Another easy way to loop through a range is to use a **For Each...Next** loop with the collection of cells returned by the **Range** method. Visual Basic automatically sets an object variable for the next cell each time the loop runs. The following procedure loops through the range A1:D10, setting to 0 (zero) any number whose absolute value is less than 0.01.

```
Sub RoundToZero2()
   For Each c In Worksheets("Sheet1").Range("A1:D10").Cells
      If Abs(c.Value) < 0.01 Then c.Value = 0
   Next
End Sub
```

If you don't know the boundaries of the range you want to loop through, you can use the **CurrentRegion** property to return the range that surrounds the active cell. For example, the following procedure, when run from a worksheet, loops through the range that surrounds the active cell, setting to 0 (zero) any number whose absolute value is less than 0.01.

```
Sub RoundToZero3()
   For Each c In ActiveCell.CurrentRegion.Cells
      If Abs(c.Value) < 0.01 Then c.Value = 0
   Next
End Sub
```

Creating a new workbook

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowCreatinganewworkbookC"}

To create a new workbook in Visual Basic, you use the **Add** method. The following procedure creates a new workbook. Microsoft Excel automatically names the workbook Book*N*, where *N* is the next available number. The new workbook becomes the active workbook.

```
Sub AddOne()
   Workbooks.Add
End Sub
```

A better way to create a new workbook is to assign it to an object variable. In the following example, the **Workbook** object returned by the **Add** method is assigned to an object variable, `newBook`. Next, several properties of `newBook` are set. You can easily control the new workbook using the object variable.

```
Sub AddNew()
Set newBook = Workbooks.Add
   With newBook
      .Title = "1995 Sales"
      .Subject = "Sales"
      .SaveAs filename:="95Sales.xls"
   End With
End Sub
```

Opening a workbook

When you open a workbook using the **Open** method, it becomes a member of the **Workbooks** collection. The following procedure opens a workbook named MyBook.xls located in the folder named "MyFolder" on drive C.

```
Sub OpenUp()
   Workbooks.Open("C:\MyFolder\MyBook.xls")
End Sub
```

Activating a workbook

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowActivatingaworkbookC"}

Activating a workbook using the **Activate** method puts the workbook in the active window. The following procedure activates the open workbook named "MyBook.xls."

```
Sub MakeActive()
   Workbooks("MyBook.xls").Activate
End Sub
```

Referring to sheets by name

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtosheetsbynameC"}

You can identify sheets by name using the **Worksheets** and **Charts** properties. The following statements activate various sheets in the active workbook.

```
Worksheets("Sheet1").Activate
Charts("Chart1").Activate

DialogSheets("Dialog1").Activate
```

You can use the **Sheets** property to return a worksheet, chart, module, or dialog sheet; the **Sheets** collection contains all of these. The following example activates the sheet named "Chart1" in the active workbook.

```
Sub ActivateChart()
   Sheets("Chart1").Activate
End Sub
```

Note Charts embedded in a worksheet are members of the **ChartObjects** collection, whereas charts that exist on their own sheets belong to the **Charts** collection.

Referring to sheets by index number

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtosheetsbyindexnumberC"}

An index number is a sequential number assigned to a sheet, based on the position of its sheet tab (counting from the left) among sheets of the same type. The following procedure uses the **Worksheets** property to activate worksheet one in the active workbook.

```
Sub FirstOne()
   Worksheets(1).Activate
End Sub
```

If you want to work with all types of sheets (worksheets, charts, modules, and dialog sheets), use the **Sheets** property. The following procedure activates sheet four in the workbook.

```
Sub FourthOne()
   Sheets(4).Activate
End Sub
```

Note The index order can change if you move, add, or delete sheets.

Referring to more than one sheet

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtomorethanonesheetC;vafctArray "}

You use the **Array** function to identify a group of sheets. The following example selects three sheets in the active workbook.

```
Sub Several()
   Worksheets(Array("Sheet1", "Sheet2", "Sheet4")).Select
End Sub
```

How to reference cells and ranges

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowHowtoreferencecellsandrangesC"}

A common task when using Visual Basic is to specify a cell or range of cells and then do something with it, such as enter a formula or change the format. You can usually do this in one statement that identifies the range and also changes a property or applies a method.

A **Range** object in Visual Basic can be either a single cell or a range of cells. The following topics show the most common ways to identify and work with **Range** objects.

Which way do you want to reference cells?

Referring to cells and ranges using A1 notation

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtocellsandrangesusingtheA1notationC"}

You can refer to a cell or range of cells in the A1 reference style by using the **Range** method. The following **Sub** procedure changes the format of cells A1:D5 to bold.

```
Sub FormatRange()
   Workbooks("Book1").Sheets("Sheet1").Range("A1:D5") _
      .Font.Bold = True
End Sub
```

The following table illustrates some A1-style references using the **Range** method.

Reference	Meaning
`Range("A1")`	Cell A1
`Range("A1:B5")`	Cells A1 through B5
`Range("C5:D9,G9:H16")`	A multiple-area selection
`Range("A:A")`	Column A
`Range("1:1")`	Row one
`Range("A:C")`	Columns A through C
`Range("1:5")`	Rows one through five
`Range("1:1,3:3,8:8")`	Rows one, three, and eight
`Range("A:A,C:C,F:F")`	Columns A, C, and F

Referring to cells using index numbers

You can use the **Cells** property to refer to a single cell by using row and column index numbers. This property returns a **Range** object that represents a single cell. In the following example, `Cells(6,1)` returns cell A6 on Sheet1. The **Value** property is then set to 10.

```
Sub EnterValue()
   Worksheets("Sheet1").Cells(6, 1).Value = 10
End Sub
```

The **Cells** property works well for looping through a range of cells, because you can substitute variables for the index numbers, as shown in the following example.

```
Sub CycleThrough()
   Dim counter As Integer
   For counter = 1 To 20
      Worksheets("Sheet1").Cells(counter, 3).Value = counter
   Next counter
End Sub
```

Note If you want to change the properties of or apply a method to a range of cells all at once, use the **Range** property. For more information, see Referring to cells using A1 notation.

Referring to rows and columns

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtorowsandcolumnsC"}

Use the **Rows** property or the **Columns** property to work with entire rows or columns. These properties return a **Range** object that represents a range of cells. In the following example, `Rows(1)` returns row one on Sheet1. The **Bold** property of the **Font** object for the range is then set to **True**.

```
Sub RowBold()
   Worksheets("Sheet1").Rows(1).Font.Bold = True
End Sub
```

The following table illustrates some row and column references using the **Rows** and **Columns** properties.

Reference	Meaning
`Rows(1)`	Row one
`Rows`	All the rows on the worksheet
`Columns(1)`	Column one
`Columns("A")`	Column one
`Columns`	All the columns on the worksheet

To work with several rows or columns at the same time, create an object variable and use the **Union** method, combining multiple calls to the **Rows** or **Columns** property. The following example changes the format of rows one, three, and five on worksheet one in the active workbook to bold.

```
Sub SeveralRows()
   Worksheets("Sheet1").Activate
   Dim myUnion As Range
   Set myUnion = Union(Rows(1), Rows(3), Rows(5))
   myUnion.Font.Bold = True
End Sub
```

Referring to cells using shortcut notation

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtocellsusingshortcutnotationC"}

You can use either the A1 reference style or a named range within brackets as a shortcut for the **Range** property. You don't have to type the word "Range" or use quotation marks, as shown in the following examples.

```
Sub ClearRange()
   Worksheets("Sheet1").[A1:B5].ClearContents
End Sub

Sub SetValue()
   [MyRange].Value = 30
End Sub
```

Referring to named ranges

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtonamedrangesC"}

Ranges are easier to identify by name than by A1 notation. To name a selected range, click the name box at the left end of the formula bar, type a name, and then press ENTER.

Referring to a named range

The following example refers to the range named "MyRange" in the workbook named "MyBook.xls."

```
Sub FormatRange()
   Range("MyBook.xls!MyRange").Font.Italic = True
End Sub
```

The following example refers to the worksheet-specific range named "Sheet1!Sales" in the workbook named "Report.xls."

```
Sub FormatSales()
   Range("[Report.xls]Sheet1!Sales").BorderAround weight:=xlthin
End Sub
```

To select a named range, use the **GoTo** method, which activates the workbook and the worksheet and then selects the range.

```
Sub ClearRange()
   Application.Goto Reference:="MyBook.xls!MyRange"
   Selection.ClearContents
End Sub
```

The following example shows how the same procedure would be written for the active workbook.

```
Sub ClearRange()
   Application.Goto Reference:="MyRange"
   Selection.ClearContents
End Sub
```

Looping through cells in a named range

The following example loops through each cell in a named range by using a **For Each...Next** loop. If the value of any cell in the range exceeds the value of `limit`, the cell color is changed to yellow.

```
Sub ApplyColor()
   Const limit As Integer = 25
   For Each c In Range("MyRange")
      If c.Value > limit Then
         c.Interior.ColorIndex = 27
      End If
   Next c
End Sub
```

Referring to cells relative to other cells

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtocellsrelativetoothercellsC"}

A common way to work with a cell relative to another cell is to use the **Offset** property. In the following example, the contents of the cell that's one row down and three columns over from the active cell on the active worksheet are formatted as double-underlined.

```
Sub Underline()
   ActiveCell.Offset(1, 3).Font.Underline = xlDouble
End Sub
```

Note You can record macros that use the **Offset** property instead of absolute references. On the **Tools** menu, point to **Record Macro**, and then click **Use Relative References**.

To loop through a range of cells, use a variable with the **Cells** property in a loop. The following example fills the first 20 cells in the third column with values between 5 and 100, incremented by 5. The variable `counter` is used as the row index for the **Cells** property.

```
Sub CycleThrough()
   Dim counter As Integer
   For counter = 1 To 20
      Worksheets("Sheet1").Cells(counter, 3).Value = counter * 5
   Next counter
End Sub
```

Referring to cells using a Range object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtocellsusingarangeobjectC;vastmDim "}

If you set an object variable to a **Range** object, you can easily manipulate the range by using the variable name.

The following procedure creates the object variable `myRange` and then assigns the variable to range A1:D5 on Sheet1 in the active workbook. Subsequent statements modify properties of the range by substituting the variable name for the range object.

```
Sub Random()
   Dim myRange As Range
   Set myRange = Worksheets("Sheet1").Range("A1:D5")
   myRange.Formula = "=RAND()"
   myRange.Font.Bold = True
End Sub
```

Referring to all the cells on the worksheet

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtoallthecellsontheworksheetC"}

When you apply the **Cells** property to a worksheet without specifying an index number, the method returns a **Range** object that represents all the cells on the worksheet. The following **Sub** procedure clears the contents from all the cells on Sheet1 in the active workbook.

```
Sub ClearSheet()
   Worksheets("Sheet1").Cells.ClearContents
End Sub
```

Referring to multiple ranges

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReferringtomultiplerangesC"}

Using the appropriate method, you can easily refer to multiple ranges. Use the **Range** and **Union** methods to refer to any group of ranges; use the **Areas** property to refer to the group of ranges selected on a worksheet.

Using the Range property

You can refer to multiple ranges with the **Range** property by putting commas between two or more references. The following example clears the contents of three ranges on Sheet1.

```
Sub ClearRanges()
   Worksheets("Sheet1").Range("C5:D9,G9:H16,B14:D18").ClearContents
End Sub
```

Named ranges make using the **Range** property to work with multiple ranges easier. The following example works when all three named ranges are on the same sheet.

```
Sub ClearNamed()
   Range("MyRange, YourRange, HisRange").ClearContents
End Sub
```

Using the Union method

You can combine multiple ranges into one **Range** object using the **Union** method. The following example creates a **Range** object called `myMultipleRange`, defines it as the ranges A1:B2 and C3:D4, and then formats the combined ranges as bold.

```
Sub MultipleRange()
   Dim r1, r2, myMultipleRange As Range
   Set r1 = Sheets("Sheet1").Range("A1:B2")
   Set r2 = Sheets("Sheet1").Range("C3:D4")
   Set myMultipleRange = Union(r1, r2)
   myMultipleRange.Font.Bold = True
End Sub
```

Using the Areas property

You can use the **Areas** property to refer to the selected range or to the collection of ranges in a multiple-area selection. The following procedure counts the areas in the selection. If there is more than one area, a warning message is displayed.

```
Sub FindMultiple()
   If Selection.Areas.Count > 1 Then
      MsgBox "Cannot do this to a multiple selection."
   End If
End Sub
```

Selecting and activating cells

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowSelectingandactivatingcellsC"}

When you work with Microsoft Excel, you usually select a cell or cells and then perform an action, such as formatting the cells or entering values in them. In Visual Basic, it is usually not necessary to select cells before modifying them.

For example, if you want to enter a formula in cell D6 using Visual Basic, you don't need to select the range D6. You just need to return the **Range** object and then set the **Formula** property to the formula you want, as shown in the following example.

```
Sub EnterFormula()
   Worksheets("Sheet1").Range("D6").Formula = "=SUM(D2:D5)"
End Sub
```

For examples of using other methods to control cells without selecting them, see How to reference cells and ranges.

Using the Select method and the Selection property

The **Select** method activates sheets and objects on sheets; the **Selection** property returns an object that represents the current selection on the active sheet in the active workbook. Before you can use the **Selection** property successfully, you must activate a workbook, activate or select a sheet, and then select a range (or other object) using the **Select** method.

The macro recorder will often create a macro that uses the **Select** method and the **Selection** property. The following **Sub** procedure was created using the macro recorder, and it illustrates how **Select** and **Selection** work together.

```
Sub Macro1()
   Sheets("Sheet1").Select
   Range("A1").Select
   ActiveCell.FormulaR1C1 = "Name"
   Range("B1").Select
   ActiveCell.FormulaR1C1 = "Address"
   Range("A1:B1").Select
   Selection.Font.Bold = True
End Sub
```

The following example accomplishes the same task without activating or selecting the worksheet or cells.

```
Sub Labels()
   With Worksheets("Sheet1")
      .Range("A1") = "Name"
      .Range("B1") = "Address"
      .Range("A1:B1").Font.Bold = True
   End With
End Sub
```

Selecting cells on the active worksheet

If you use the **Select** method to select cells, be aware that **Select** works only on the active worksheet. If you run your **Sub** procedure from the module, the **Select** method will fail unless your procedure activates the worksheet before using the **Select** method on a range of cells. For example, the following procedure copies a row from Sheet1 to Sheet2 in the active workbook.

```
Sub CopyRow()
   Worksheets("Sheet1").Rows(1).Copy
   Worksheets("Sheet2").Select
```

```
   Worksheets("Sheet2").Rows(1).Select
   Worksheets("Sheet2").Paste
End Sub
```

Activating a cell within a selection

You can use the **Activate** method to activate a cell within a selection. There can be only one active cell, even when a range of cells is selected. The following procedure selects a range and then activates a cell within the range without changing the selection.

```
Sub MakeActive()
   Worksheets("Sheet1").Activate
   Range("A1:D4").Select
   Range("B2").Activate
End Sub
```

Working with the active cell

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowWorkingwiththeactivecellC"}

The **ActiveCell** property returns a **Range** object that represents the cell that is active. You can apply any of the properties or methods of a **Range** object to the active cell, as in the following example.

```
Sub SetValue()
   Worksheets("Sheet1").Activate
   ActiveCell.Value = 35
End Sub
```

Note You can work with the active cell only when the worksheet that it is on is the active sheet.

Moving the active cell

You can use the **Activate** method to designate which cell is the active cell. For example, the following procedure makes B5 the active cell and then formats it as bold.

```
Sub SetActive()
   Worksheets("Sheet1").Activate
   Worksheets("Sheet1").Range("B5").Activate
   ActiveCell.Font.Bold = True
End Sub
```

Note To select a range of cells, use the **Select** method. To make a single cell the active cell, use the **Activate** method.

You can use the **Offset** property to move the the active cell. The following procedure inserts text into the active cell in the selected range and then moves the active cell one cell to the right without changing the selection.

```
Sub MoveActive()
   Worksheets("Sheet1").Activate
   Range("A1:D10").Select
   ActiveCell.Value = "Monthly Totals"
   ActiveCell.Offset(0, 1).Activate
End Sub
```

Selecting the cells surrounding the active cell

The **CurrentRegion** property returns a range of cells bounded by blank rows and columns. In the following example, the selection is expanded to include the cells adjoining the active cell that contain data. This range is then formatted with the Currency style.

```
Sub Region()
   Worksheets("Sheet1").Activate
   ActiveCell.CurrentRegion.Select
   Selection.Style = "Currency"
End Sub
```

Working with 3-D ranges

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowWorkingwith3DrangesC;vafctArray "}

If you are working with the same range on more than one sheet, use the **Array** function to specify two or more sheets to select. The following example formats the border of a 3-D range of cells.

```
Sub FormatSheets()
   Sheets(Array("Sheet2", "Sheet3", "Sheet5")).Select
   Range("A1:H1").Select
   Selection.Borders(xlBottom).LineStyle = xlDouble
End Sub
```

The following example applies the **FillAcrossSheets** method to transfer the formats and any data from the range on Sheet2 to the corresponding ranges on all the worksheets in the active workbook.

```
Sub FillAll()
   Worksheets("Sheet2").Range("A1:H1") _
      .Borders(xlBottom).LineStyle = xlDouble
   Worksheets.FillAcrossSheets (Worksheets("Sheet2") _
      .Range("A1:H1"))
End Sub
```

Working with shapes (drawing objects)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowWorkingWithShapesC"}

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 format or 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.

```
Worksheets(1).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.

Creating a ShapeRange collection that contains all shapes on a sheet

You can create a **ShapeRange** object that contains all the **Shape** objects on a sheet by selecting the shapes and then using the **ShapeRange** property to return a **ShapeRange** object containing the selected shapes.

```
Worksheets(1).Shapes.Select
Set sr = Selection.ShapeRange
```

In Microsoft Excel, the ***Index*** argument is not optional for the **Range** property of the **Shapes** collection, so you cannot use this property without an argument to create a **ShapeRange** object containing all shapes in a **Shapes** collection.

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 shapes named "Big Star" and "Little Star" on `myDocument` and applies a gradient fill to them.

```
Set myDocument = Worksheets(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 a **Shapes** or **ShapeRange** collection that contains the shapes you want to work with, 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 changes the foreground color for each shape that is an AutoShape.

```
Set myDocument = Worksheets(1)
For Each sh In myDocument.Shapes
    If sh.Type = msoAutoShape Then
        sh.Fill.ForeColor.RGB = RGB(255, 0, 0)
    End If
Next
```

The following example constructs a **ShapeRange** collection that contains all the currently selected shapes in the active window and sets the foreground color for each selected shape.

```
For Each sh in ActiveWindow.Selection.ShapeRange
    sh.Fill.ForeColor.RGB = RGB(255, 0, 0)
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":"xlhowControllingAppsC"}

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 sheets

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowUsingActiveXControlsSheetsC"}

This topic covers specific information about using ActiveX controls on worksheets and chart sheets. 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 sheets.

- In addition to the standard properties available for ActiveX controls, the following properties can be used with ActiveX controls in Microsoft Excel: **BottomRightCell**, **LinkedCell**, **ListFillRange**, **Placement**, **PrintObject**, **TopLeftCell**, and **ZOrder**.

  These properties can be set and returned using the ActiveX control name. The following example scrolls the workbook window so CommandButton1 is in the upper-left corner.

  ```
  Set t = Sheet1.CommandButton1.TopLeftCell
  With ActiveWindow
     .ScrollRow = t.Row
     .ScrollColumn = t.Column
  End With
  ```

- Some Microsoft Excel Visual Basic methods and properties are disabled when an ActiveX control is activated. For example, the **Sort** method cannot be used when a control is active, so the following code fails in a button click event procedure (because the control is still active after the user clicks it).

  ```
  Private Sub CommandButton1.Click
     Range("a1:a10").Sort Key1:=Range("a1")
  End Sub
  ```

  You can work around this problem by activating some other element on the sheet before you use the properety or method that failed. For example, the following code sorts the range:

  ```
  Private Sub CommandButton1.Click
     Range("a1").Activate
     Range("a1:a10").Sort Key1:=Range("a1")
     CommandButton1.Activate
  End Sub
  ```

- Controls on a Microsoft Excel workbook embedded in a document in another application will not work if the user double clicks the workbook to edit it. The controls will work if the user right clicks the workbook and selects the **Open** command from the shortcut menu.
- When a Microsoft Excel 97 workbook is saved using the Microsoft Excel 5.0/95 Workbook file format, ActiveX control information is lost.
- The **Me** keyword in an event procedure for an ActiveX control on a sheet refers to the sheet, not to the control.

Adding Controls With Visual Basic

In Microsoft Excel, ActiveX controls are represented by **OLEObject** objects in the **OLEObjects** collection (all **OLEObject** objects are also in the **Shapes** collection). To programmatically add an ActiveX control to a sheet, use the **Add** method of the **OLEObjects** collection. The following example adds a command button to worksheet one.

```
Worksheets(1).OLEObjects.Add "Forms.CommandButton.1", _
   Left:=10, Top:=10, Height:=20, Width:=100
```

Using Control Properties With Visual Basic

Most often, your Visual Basic code will refer to ActiveX controls by name. The following example changes the caption on the control named "CommandButton1."

```
Sheet1.CommandButton1.Caption = "Run"
```

Note that when you use a control name outside the class module for the sheet containing the control, you must qualify the control name with the sheet name.

To change the control name you use in Visual Basic code, select the control and set the **(Name)** property in the Properties window.

Because ActiveX controls are also represented by **OLEObject** objects in the **OLEObjects** collection, you can set control properties using the objects in the collection. The following example sets the left position of the control named "CommandButton1."

```
Worksheets(1).OLEObjects("CommandButton1").Left = 10
```

Control properties that are not shown as properties of the **OLEObject** object can be set by returning the actual control object using the **Object** property. The following example sets the caption for CommandButton1.

```
Worksheets(1).OLEObjects("CommandButton1").Object.Caption = "run me"
```

Because all OLE objects are also members of the **Shapes** collection, you can use the collection to set properties for several controls. The following example aligns the left edge of all controls on worksheet one.

```
For Each s In Worksheets(1).Shapes
   If s.Type = msoOLEControlObject Then s.Left = 10
Next
```

Using ActiveX controls on a document

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowUsingControlsC"}

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 Microsoft Excel, see Using ActiveX controls on sheets.

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":"xlhowCreatingaCustomDialogBoxC"}

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":"xlhowCreatingAUserFormC"}

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":"xlhowAddingControlstoaUserFormC"}

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":"xlhowAddingControlstoaDocumentC"}

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":"xlhowSettingControlPropertiesC"}

You can set some control properties at design time (before any macro is running). In design mode, right-click a control and click **Properties** to display the Properties window. Property names are shown in the left column in the window, property values in the right column. You set a property value by entering the new value to the right of the property name.

Initializing control properties

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowInitializingControlPropertiesC"}

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()
   UserForm1.lstNames.AddItem "Test One"
   UserForm1.lstNames.AddItem "Test Two"
   UserForm1.txtUserName.Text = "Default Name"
End Sub
```

Control and dialog box events

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowControlandDialogBoxEventsC"}

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":"xlhowDisplayingaDialogBoxC"}

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":"xlhowSetControlPropertiesDynamicC"}

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
```

design And run modes

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.

Using events with Microsoft Excel objects

{ewc HLP95EN.DLL, DYNALINK, "See Also":"UsingEventswithMicrosoftExcelObjectsC"}

You can write event procedures in Microsoft Excel at the worksheet, chart, workbook, or application level. For example, the **Activate** event occurs at the sheet level, and the **SheetActivate** event is available at both the workbook and application levels The **SheetActivate** event for a workbook occurs when any sheet in the workbook is activated. At the application level, the **SheetActivate** event occurs when any sheet in any open workbook is activated.

Worksheet, chart sheet, and workbook event procedures are available for any open sheet or workbook. To write event procedures for an embedded chart or for the **Application** object, you must create a new object using the **WithEvents** keyword in a class module.

Use the **EnableEvents** property to enable or disable events. For example, using the **Save** method to save a workbook causes the **BeforeSave** event to occur. You can prevent this by setting the **EnableEvents** property to **False** before you call the **Save** method.

```
Application.EnableEvents = False
ActiveWorkbook.Save
Application.EnableEvents = True
```

Worksheet object events

{ewc HLP95EN.DLL, DYNALINK, "See Also":"WorksheetEventsC"}

Events on sheets are enabled by default. To view the event procedures for a sheet, right-click the sheet tab and click **View Code** on the shortcut menu. Select the event name from the **Procedure** drop-down list box.

Activate
BeforeDoubleClick
BeforeRightClick
Calculate
Change
Deactivate
SelectionChange

Worksheet-level events occur when a worksheet is activated or the user changes a worksheet cell. The following example adjusts the size of columns A through F whenever the worksheet is recalculated.

```
Private Sub Worksheet_Calculate()
   Columns("A:F").AutoFit
End Sub
```

Some events can be used to substitute an action for the default application behavior, or to make a small change to the default behavior. The following example traps the right-click event and adds a new menu item to the shortcut menu for cells B1:B10.

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, _
        Cancel As Boolean)
    For Each icbc In Application.CommandBars("cell").Controls
        If icbc.Tag = "brccm" Then icbc.Delete
    Next icbc
    If Not Application.Intersect(Target, Range("b1:b10")) Is Nothing Then
            With Application.CommandBars("cell").Controls _
                .Add(Type:=msoControlButton, before:=6, _
                temporary:=True)
            .Caption = "New Context Menu Item"
            .OnAction = "MyMacro"
            .Tag = "brccm"
        End With
    End If
End Sub
```

Chart object events

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ChartEventsC"}

Chart events occur when the user activates or changes a chart. Events on chart sheets are enabled by default. To view the event procedures for a sheet, right-click the sheet tab and select **View Code** from the shortcut menu. Select the event name from the **Procedure** drop-down list box.

Activate
BeforeDoubleClick
BeforeRightClick
Calculate
Deactivate
DragOver
DragPlot
MouseDown
MouseMove
MouseUp
Resize
Select
SeriesChange

Note To write event procedures for an embedded chart, you must create a new object using the **WithEvents** keyword in a class module. For more information, see Using Events with Embedded Charts.

This example changes a point's border color when the user changes the point value.

```
Private Sub Chart_SeriesChange(ByVal SeriesIndex As Long, _
      ByVal PointIndex As Long)
    Set p = ActiveChart.SeriesCollection(SeriesIndex).Points(PointIndex)
    p.Border.ColorIndex = 3
End Sub
```

Workbook object events

{ewc HLP95EN.DLL, DYNALINK, "See Also":"WorkbookEventsC"}

Workbook events occur when the workbook changes or when any sheet in the workbook changes. Events on workbooks are enabled by default. To view the event procedures for a workbook, right-click the title bar of a restored or minimized workbook window and click **View Code** on the shortcut menu. Select the event name from the **Procedure** drop-down list box.

AddinUninstall
BeforeClose
BeforePrint
BeforeSave
Deactivate
NewSheet
Open
SheetActivate
SheetBeforeDoubleClick
SheetBeforeRightClick
SheetCalculate
SheetChange
SheetDeactivate
SheetSelectionChange
WindowActivate
WindowDeactivate
WindowResize

This example maximizes Microsoft Excel when the workbook is opened

```
Sub Workbook_Open()
   Application.WindowState = xlMaximized
End Sub
```

Application object events

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ApplicationEventsC"}

Application events occur when a workbook is created or opened or when any sheet in any open workbook changes. To write event procedures for the Application object, you must create a new object using the **WithEvents** keyword in a class module. For more information, see Using Events with the Application Object.

NewWorkbook
SheetActivate
SheetBeforeDoubleClick
SheetBeforeRightClick
SheetCalculate
SheetChange
SheetDeactivate
SheetSelectionChange
WindowActivate
WindowDeactivate
WindowResize
WorkbookActivate
WorkbookAddinInstall
WorkbookAddinUninstall
WorkbookBeforeClose
WorkbookBeforePrint
WorkbookBeforeSave
WorkbookDeactivate
WorkbookNewSheet
WorkbookOpen

AddIns Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAddInsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAddInsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAddInsA "}

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.

Remarks

Using this method without an object qualifier is equivalent to `Application.Addins`.

AddIns Property Example

This example displays the status of the Analysis ToolPak add-in. Note that the string used as the index to the **AddIns** collection is the title of the add-in, not the add-in's file name.

```
If AddIns("Analysis ToolPak").Installed = True Then
   MsgBox "Analysis ToolPak add-in is installed"
Else
   MsgBox "Analysis ToolPak add-in is not installed"
End If
```

Areas Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAreasC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAreasX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAreasA "}

Returns an **Areas** collection that represents all the ranges in a multiple-area selection. Read-only.

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

Remarks

For a single selection, the **Areas** property returns a collection that contains one object – the original **Range** object itself. For a multiple-area selection, the **Areas** property returns a collection that contains one object for each selected area.

Areas Property Example

This example displays a message if the user tries to carry out a command when more than one area is selected. This example must be run from a worksheet.

```
If Selection.Areas.Count > 1 Then
   MsgBox "Cannot do this to a multi-area selection."
End If
```

Columns Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColumnsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColumnsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColumnsA "}

Application object: Returns a **Range** object that represents all the columns on the active worksheet. If the active document isn't a worksheet, the **Columns** property fails. Read-only.

Range object: Returns a **Range** object that represents the columns in the specified range. Read-only.

Worksheet object: Returns a **Range** object that represents all the columns on the specified worksheet. Read-only.

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

Remarks

Using this property without an object qualifier is equivalent to using `ActiveSheet.Columns`.

When applied to a **Range** object that's a multiple-area selection, this property returns columns from only the first area of the range. For example, if the **Range** object has two areas – A1:B2 and C3:D4 – `Selection.Columns.Count` returns 2, not 4. To use this property on a range that may contain a multiple-area selection, test `Areas.Count` to determine whether the range contains more than one area. If it does, loop over each area in the range.

Columns Property Example

This example formats the font of column one (column A) on Sheet1 as bold.

```
Worksheets("Sheet1").Columns(1).Font.Bold = True
```

This example sets the value of every cell in column one in the range named "myRange" to 0 (zero).

```
Range("myRange").Columns(1).Value = 0
```

This example displays the number of columns in the selection on Sheet1. If more than one area is selected, the example loops through each area.

```
Worksheets("Sheet1").Activate
areaCount = Selection.Areas.Count
If areaCount <= 1 Then
   MsgBox "The selection contains " & _
      Selection.Columns.Count & " columns."
Else
   For i = 1 To areaCount
      MsgBox "Area " & i & " of the selection contains " & _
         Selection.Areas(i).Columns.Count & " columns."
   Next i
End If
```

Dialogs Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDialogsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDialogsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDialogsA "}

Returns a **Dialogs** collection that represents all built-in dialog boxes. Read-only.

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

Dialogs Property Example

This example displays the **Open** dialog box (**File** menu).

```
Application.Dialogs(xlDialogOpen).Show
```

Excel4IntlMacroSheets Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproExcel4IntlMacroSheetsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproExcel4IntlMacroSheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproExcel4IntlMacroSheetsA "}

Returns a **Sheets** collection that represents all the Microsoft Excel 4.0 international macro sheets in the specified workbook. Read-only.

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

Remarks

Using this property with the **Application** object or without an object qualifier is equivalent to using `ActiveWorkbook.Excel4IntlMacroSheets`.

Excel4IntlMacroSheets Property Example

This example displays the number of Microsoft Excel 4.0 international macro sheets in the active workbook.

```
MsgBox "There are " & ActiveWorkbook.Excel4IntlMacroSheets.Count & _
   " Microsoft Excel 4.0 international macro sheets in this workbook."
```

Excel4MacroSheets Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproExcel4MacroSheetsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproExcel4MacroSheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproExcel4MacroSheetsA "}

Returns a **Sheets** collection that represents all the Microsoft Excel 4.0 macro sheets in the specified workbook. Read-only.

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

Remarks

Using this property with the **Application** object or without an object qualifier is equivalent to using `ActiveWorkbook.Excel4MacroSheets`.

Excel4MacroSheets Property Example

This example displays the number of Microsoft Excel 4.0 macro sheets in the active workbook.

```
MsgBox "There are " & ActiveWorkbook.Excel4MacroSheets.Count & _
   " Microsoft Excel 4.0 macro sheets in this workbook."
```

Names Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNamesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNamesA "}

Application object: Returns a **Names** collection that represents all the names in the active workbook. Read-only.

Workbook object: Returns a **Names** collection that represents all the names in the specified workbook (including all worksheet-specific names). Read-only.

Worksheet object: Returns a **Names** collection that represents all the worksheet-specific names (names defined with the "WorksheetName!" prefix). Read-only.

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

Remarks

Using this property without an object qualifier is equivalent to using `ActiveWorkbook.Names`.

Names Property Example

This example defines the name "myName" for cell A1 on Sheet1.

```
ActiveWorkbook.Names.Add Name:="myName", RefersToR1C1:= _
    "=Sheet1!R1C1"
```

Offset Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOffsetC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOffsetExampleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOffsetA "}

Returns a **Range** object that represents a range that's offset from the specified range. Read-only.

Syntax

expression.**Offset(*RowOffset, ColumnOffset*)**

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

RowOffset Optional **Variant**. The number of rows (positive, negative, or zero) by which the range is to be offset. The default value is 0 (zero).

ColumnOffset Optional **Variant**. The number of columns (positive, negative, or zero) by which the range is to be offset. The default value is 0 (zero).

Offset Property Example

This example activates the cell three columns to the right of and three rows down from the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveCell.Offset(rowOffset:=3, columnOffset:=3).Activate
```

This example assumes that Sheet1 contains a table that has a header row. The example selects the table, without selecting the header row. The active cell must be somewhere in the table before the example is run.

```
Set tbl = ActiveCell.CurrentRegion
tbl.Offset(1, 0).Resize(tbl.Rows.Count - 1, tbl.Columns.Count).Select
```

Panes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPanesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPanesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPanesA "}

Returns a **Panes** collection that represents all the panes in the specified window. Read-only.

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

Remarks

This property is available for a window only if the window's **Split** property can be set to **True**.

Panes Property Example

This example displays the number of panes in the active window in Book1.xls.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
MsgBox "There are " & ActiveWindow.Panes.Count & _
   " panes in the active window"
```

This example activates the pane in the upper-left corner of the active window in Book1.xls.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.Panes(1).Activate
```

PreviousSelections Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPreviousSelectionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPreviousSelectionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPreviousSelectionsA "}

Returns an array of the last four ranges or names selected. Each element in the array is a **Range** object. Read-only **Variant**.

Syntax

expression.**PreviousSelections(*Index*)**

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

Index Optional **Variant**. The index number (from 1 to 4) of the previous range or name.

Remarks

Each time you go to a range or cell by using the **Name** box or the **Go To** command (**Edit** menu), or each time a macro calls the **Goto** method, the previous range is added to this array as element number 1, and the other items in the array are moved down.

PreviousSelections Property Example

This example displays the cell addresses of all items in the array of previous selections. If there are no previous selections, the **LBound** function returns an error. This error is trapped, and a message box appears.

```
On Error GoTo noSelections
For i = LBound(Application.PreviousSelections) To _
         UBound(Application.PreviousSelections)
   MsgBox Application.PreviousSelections(i).Address
Next i
Exit Sub
On Error GoTo 0

noSelections:
   MsgBox "There are no previous selections"
```

Rows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowsA "}

Application object: Returns a **Range** object that represents all the rows on the active worksheet. If the active document isn't a worksheet, the **Rows** property fails. Read-only.

Range object: Returns a **Range** object that represents the rows in the specified range. Read-only.

Worksheet object: Returns a **Range** object that represents all the rows on the specified worksheet. Read-only.

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

Remarks

Using this property without an object qualifier is equivalent to using `ActiveSheet.Rows`.

When applied to a **Range** object that's a multiple selection, this property returns rows from only the first area of the range. For example, if the **Range** object has two areas – A1:B2 and C3:D4 – `Selection.Rows.Count` returns 2, not 4. To use this property on a range that may contain a multiple selection, test `Areas.Count` to determine whether the range is a multiple selection. If it is, loop over each area in the range, as shown in the third example.

Rows Property Example

This example deletes row three on Sheet1.

```
Worksheets("Sheet1").Rows(3).Delete
```

This example deletes rows in the current region on worksheet one where the value of cell one in the row is the same as the value in cell one in the previous row.

```
For Each rw In Worksheets(1).Cells(1, 1).CurrentRegion.Rows
   this = rw.Cells(1, 1).Value
   If this = last Then rw.Delete
   last = this
Next
```

This example displays the number of rows in the selection on Sheet1. If more than one area is selected, the example loops through each area.

```
Worksheets("Sheet1").Activate
areaCount = Selection.Areas.Count
If areaCount <= 1 Then
   MsgBox "The selection contains " & _
      Selection.Rows.Count & " rows."
Else
   i = 1
   For Each a In Selection.Areas
      MsgBox "Area " & i & " of the selection contains " & _
         a.Rows.Count & " rows."
      i = i + 1
   Next a
End If
```

SelectedSheets Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSelectedSheetsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSelectedSheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSelectedSheetsA "}

Returns a **Sheets** collection that represents all the selected sheets in the specified window. Read-only.

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

SelectedSheets Property Example

This example displays a message if Sheet1 is selected in Book1.xls.

```
For Each sh In Workbooks("BOOK1.XLS").Windows(1).SelectedSheets
   If sh.Name = "Sheet1" Then
      MsgBox "Sheet1 is selected"
      Exit For
   End If
Next
```

Sheets Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSheetsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSheetsA "}

Application object: Returns a **Sheets** collection that represents all the sheets in the active workbook. Read-only.

Workbook object: Returns a **Sheets** collection that represents all the sheets in the specified workbook. Read-only.

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

Remarks

Using this property without an object qualifier is equivalent to using `ActiveWorkbook.Sheets`.

Sheets Property Example

This example creates a new worksheet and then places a list of the active workbook's sheet names in the first column.

```
Set newSheet = Sheets.Add(Type:=xlWorksheet)
For i = 1 To Sheets.Count
   newSheet.Cells(i, 1).Value = Sheets(i).Name
Next i
```

Styles Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStylesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStylesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStylesA "}

Returns a **Styles** collection that represents all the styles in the specified workbook. Read-only.

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

Styles Property Example

This example deletes the user-defined style "Stock Quote Style" from the active workbook.

```
ActiveWorkbook.Styles("Stock Quote Style").Delete
```

Windows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWindowsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWindowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWindowsA "}

Application object: Returns a **Windows** collection that represents all the windows in all the workbooks. Read-only.

Workbook object: Returns a **Windows** collection that represents all the windows in the specified workbook. Read-only.

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

Remarks

Using this property without an object qualifier is equivalent to using `Application.Windows`.

This property returns a collection of both visible and hidden windows.

Windows Property Example

This example closes the first open or hidden window in Microsoft Excel.

```
Application.Windows(1).Close
```

This example names window one in the active workbook "Consolidated Balance Sheet." This name is then used as the index to the **Windows** collection.

```
ActiveWorkbook.Windows(1).Caption = "Consolidated Balance Sheet"
ActiveWorkbook.Windows("Consolidated Balance Sheet") _
    .ActiveSheet.Calculate
```

Workbooks Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWorkbooksC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWorkbooksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWorkbooksA "}

Returns a **Workbooks** collection that represents all the open workbooks. Read-only.

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

Remarks

Using this property without an object qualifier is equivalent to using `Application.Workbooks`.

The collection returned by the **Workbooks** property doesn't include open add-ins, which are a special kind of hidden workbook. You can, however, return a single open add-in if you know the file name. For example, `Workbooks("Oscar.xla")` will return the open add-in named "Oscar.xla" as a **Workbook** object.

Workbooks Property Example

This example activates the workbook Book1.xls.

```
Workbooks("BOOK1").Activate
```

This example opens the workbook Large.xls.

```
Workbooks.Open filename:="LARGE.XLS"
```

This example saves changes to and closes all workbooks except the one that's running the example.

```
For Each w In Workbooks
   If w.Name <> ThisWorkbook.Name Then
      w.Close savechanges:=True
   End If
Next w
```

ActiveCell Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproActiveCellC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproActiveCellX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproActiveCellA "}

Returns a **Range** object that represents the active cell in the active window (the window on top) or in the specified window. If the window isn't displaying a worksheet, this property fails. Read-only.

Remarks

If you don't specify an object qualifier, this property returns the active cell in the active window.

Be careful to distinguish between the active cell and the selection. The active cell is a single cell inside the current selection. The selection may contain more than one cell, but only one is the active cell.

The following expressions all return the active cell, and are all equivalent.

```
ActiveCell
Application.ActiveCell
ActiveWindow.ActiveCell
Application.ActiveWindow.ActiveCell
```

ActiveCell Property Example

This example uses a message box to display the value in the active cell. Because the **ActiveCell** property fails if the active sheet isn't a worksheet, the example activates Sheet1 before using the **ActiveCell** property.

```
Worksheets("Sheet1").Activate
MsgBox ActiveCell.Value
```

This example changes the font formatting for the active cell.

```
Worksheets("Sheet1").Activate
With ActiveCell.Font
    .Bold = True
    .Italic = True
End With
```

ActivePrinter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproActivePrinterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproActivePrinterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproActivePrinterA "}

Returns or sets the name of the active printer. Read/write **String**.

Remarks

This property cannot be set on the Apple Macintosh.

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":"xlproActiveWindowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproActiveWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproActiveWindowA "}

Returns a **Window** object that represents the active window (the window on top). Read-only. Returns **Nothing** if there are no windows open.

ActiveWindow Property Example

This example displays the name (**Caption** property) of the active window.

```
MsgBox "The name of the active window is " & ActiveWindow.Caption
```

ActiveWorkbook Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproActiveWorkbookC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproActiveWorkbookX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproActiveWorkbookA "}

Returns a **Workbook** object that represents the workbook in the active window (the window on top). Read-only. Returns **Nothing** if there are no windows open or if either the Info window or the Clipboard window is the active window.

ActiveWorkbook Property Example

This example displays the name of the active workbook.

```
MsgBox "The name of the active workbook is " & ActiveWorkbook.Name
```

AltStartupPath Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAltStartupPathC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAltStartupPathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAltStartupPathA "}

Returns or sets the name of the alternate startup folder. Read/write **String**.

AltStartupPath Property Example

This example sets the alternate startup folder.

```
Application.AltStartupPath = "C:\EXCEL\MACROS"
```

This is the same example in Microsoft Excel for the Macintosh.

```
Application.AltStartupPath = "HD:Excel:My Macros"
```

Application Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproApplicationC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproApplicationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproApplicationA "}

Used *without* an object qualifier, this property returns an **Application** object that represents the Microsoft Excel application. Used *with* an object qualifier, this property returns an **Application** object that represents the creator of the specified object (you can use this property with an OLE Automation object to return that object's application). Read-only.

Application Property Example

This example displays a message about the application that created `myObject`.

```
Set myObject = ActiveWorkbook
If myObject.Application.Value = "Microsoft Excel" Then
   MsgBox "This is a Microsoft Excel object"
Else
   MsgBox "This is not a Microsoft Excel object"
End If
```

Calculate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCalculateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCalculateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCalculateA "}

Calculates all open workbooks, a specific worksheet in a workbook, or a specified range of cells on a worksheet, as shown in the following table.

To calculate	Follow this example
All open workbooks	`Application.Calculate` (or just `Calculate`)
A specific worksheet	`Worksheets(1).Calculate`
A specified range	`Worksheets(1).Rows(2).Calculate`

Syntax

expression.**Calculate**

expression Optional for **Application**, required for **Worksheet** and **Range**. An expression that returns an object in the Applies To list.

Calculate Method Example

This example calculates the formulas in columns A, B, and C in the used range on Sheet1.

```
Worksheets("Sheet1").UsedRange.Columns("A:C").Calculate
```

CalculateBeforeSave Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCalculateBeforeSaveC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCalculateBeforeSaveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCalculateBeforeSaveA "}

True if workbooks are calculated before they're saved to disk (if the **Calculation** property is set to **xlManual**). This property is preserved even if you change the **Calculation** property. Read/write **Boolean**.

CalculateBeforeSave Property Example

This example sets Microsoft Excel to calculate workbooks before they're saved to disk.

```
Application.Calculation = xlManual
Application.CalculateBeforeSave = True
```

CanPlaySounds Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCanPlaySoundsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCanPlaySoundsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCanPlaySoundsA "}

This property should not be used. Sound notes have been removed from Microsoft Excel.

CanRecordSounds Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCanRecordSoundsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCanRecordSoundsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCanRecordSoundsA "}

This property should not be used. Sound notes have been removed from Microsoft Excel.

CellDragAndDrop Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCellDragAndDropC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCellDragAndDropX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCellDragAndDropA "}

True if dragging and dropping cells is enabled. Read/write **Boolean**.

CellDragAndDrop Property Example

This example enables dragging and dropping cells.

```
Application.CellDragAndDrop = True
```

CommandUnderlines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCommandUnderlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCommandUnderlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCommandUnderlinesA "}

Returns or sets the state of the command underlines in Microsoft Excel for the Macintosh. Can be one of the following **XlCommandUnderlines** constants: **xlCommandUnderlinesOn**, **xlCommandUnderlinesOff**, or **xlCommandUnderlinesAutomatic**. Read/write **Long**.

Remarks

In Microsoft Excel for Windows, reading this property always returns **xlCommandUnderlinesOn**, and setting this property to anything other than **xlCommandUnderlinesOn** is an error.

CommandUnderlines Property Example

This example turns off command underlines in Microsoft Excel for the Macintosh.

```
Application.CommandUnderlines = xlCommandUnderlinesOff
```

ConstrainNumeric Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproConstrainNumericC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConstrainNumericX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConstrainNumericA "}

True if handwriting recognition is limited to numbers and punctuation only. Read/write **Boolean**.

Note This property is available only if you're using Microsoft Windows for Pen Computing. If you try to set this property under any other operating system, an error occurs.

ConstrainNumeric Property Example

This example limits handwriting recognition to numbers and punctuation only if Microsoft Windows for Pen Computing is running.

```
If Application.WindowsForPens Then
   Application.ConstrainNumeric = True
End If
```

ConvertFormula Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthConvertFormulaC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthConvertFormulaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthConvertFormulaA "}

Converts cell references in a formula between the A1 and R1C1 reference styles, between relative and absolute references, or both.

Syntax

expression.**ConvertFormula(*Formula, FromReferenceStyle, ToReferenceStyle, ToAbsolute, RelativeTo*)**

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

Formula Required **Variant**. A string that containis the formula you want to convert. This must be a valid formula, and it must begin with an equal sign.

FromReferenceStyle Required **Long**. The reference style of the formula. Can be one of the following **XLReferenceStyle** constants: **xlA1** or **xlR1C1**.

ToReferenceStyle Optional **Variant**. The reference style you want returned. Can be one of the following **XLReferenceStyle** constants: **xlA1** or **xlR1C1**. If this argument is omitted, the reference style isn't changed; the formula stays in the style specified by ***FromReferenceStyle***.

ToAbsolute Optional **Variant**. Specifies the converted reference type. Can be one of the following **XLReferenceType** constants: **xlAbsolute**, **xlAbsRowRelColumn**, **xlRelRowAbsColumn**, or **xlRelative**. If this argument is omitted, the reference type isn't changed.

RelativeTo Optional **Variant**. A **Range** object that contains one cell. Relative references relate to this cell.

ConvertFormula Method Example

This example converts a SUM formula that contains R1C1-style references to an equivalent formula that contains A1-style references, and then it displays the result.

```
inputFormula = "=SUM(R10C2:R15C2)"
MsgBox Application.ConvertFormula( _
   formula:=inputFormula, _
   fromReferenceStyle:=xlR1C1, _
   toReferenceStyle:=xlA1)
```

Creator Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCreatorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCreatorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCreatorA "}

Returns a 32-bit integer that indicates the application in which this object was created. If the object was created in Microsoft Excel, this property returns the string XCEL, which is equivalent to the hexadecimal number 5843454C. Read-only **Long**.

Remarks

The **Creator** property is designed to be used in Microsoft Excel for the Macintosh, where each application has a four-character creator code. For example, Microsoft Excel has the creator code XCEL.

Creator Property Example

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

```
Set myObject = ActiveWorkbook
If myObject.Creator = &h5843454c Then
   MsgBox "This is a Microsoft Excel object"
Else
   MsgBox "This is not a Microsoft Excel object"
End If
```

CutCopyMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCutCopyModeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCutCopyModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCutCopyModeA "}

Returns or sets the status of Cut or Copy mode. Can be **True**, **False**, or a **XLCutCopyMode** constant, as shown in the following tables. Read/write **Long**.

Return value	Description
False	Not in Cut or Copy mode
xlCopy	In Copy mode
xlCut	In Cut mode

Set value	Description
False	Cancels Cut or Copy mode and removes the moving border.
True	Cancels Cut or Copy mode and removes the moving border. On the Macintosh, this also places the contents of the selection on the Macintosh Clipboard.

CutCopyMode Property Example

This example uses a message box to display the status of Cut or Copy mode.

```
Select Case Application.CutCopyMode
   Case Is = False
      MsgBox "Not in Cut or Copy mode"
   Case Is = xlCopy
      MsgBox "In Copy mode"
   Case Is = xlCut
      MsgBox "In Cut mode"
End Select
```

DataEntryMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataEntryModeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataEntryModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataEntryModeA "}

Returns or sets Data Entry mode, as shown in the following table. When in Data Entry mode, you can enter data only in the unlocked cells in the currently selected range. Read/write **Long**.

Value	Meaning
xlOn	Data Entry mode is turned on.
xlOff	Data Entry mode is turned off.
xlStrict	Data Entry mode is turned on, and pressing ESC won't turn it off.

DataEntryMode Property Example

This example turns off Data Entry mode if it's on.

```
If (Application.DataEntryMode = xlOn) Or _
   (Application.DataEntryMode = xlStrict) Then
      Application.DataEntryMode = xlOff
End If
```

DDEAppReturnCode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDDEAppReturnCodeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDDEAppReturnCodeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDDEAppReturnCodeA "}

Returns the application-specific DDE return code that was contained in the last DDE acknowledge message received by Microsoft Excel. Read-only **Long**.

DDEAppReturnCode Property Example

This example sets the variable `appErrorCode` to the DDE return code.

```
appErrorCode = Application.DDEAppReturnCode
```

DDEExecute Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDDEExecuteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDDEExecuteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDDEExecuteA "}

Runs a command or performs some other action or actions in another application by way of the specified DDE channel.

Syntax

expression.**DDEExecute(*Channel, String*)**

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

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

String Required **String**. The message defined in the receiving application.

Remarks

The **DDEExecute** method is designed to send commands to another application. You can also use it to send keystrokes to another application, although the **SendKeys** method is the preferred way to send keystrokes. The ***String*** argument can specify any single key: in Windows, any key combined with ALT, CTRL, or SHIFT, or any combination of those keys; or on the Macintosh, any key combined with COMMAND, CTRL, OPTION, or SHIFT, or any combination of those keys. Each key is represented by one or more characters, such as `"a"` for the character a, or `"{ENTER}"` for the ENTER key.

To specify characters that aren't displayed when you press the corresponding key (for example, ENTER or TAB), use the codes listed in the following table. Each code in the table represents one key on the keyboard.

Key	Code
BACKSPACE	`{BACKSPACE}` or `{BS}`
BREAK	`{BREAK}`
CAPS LOCK	`{CAPSLOCK}`
CLEAR	`{CLEAR}`
DELETE or DEL	`{DELETE}` or `{DEL}`
DOWN ARROW	`{DOWN}`
END	`{END}`
ENTER (numeric keypad)	`{ENTER}`
ENTER	~ (tilde)
ESC	`{ESCAPE}` or `{ESC}`
HELP	`{HELP}`
HOME	`{HOME}`
INS	`{INSERT}`
LEFT ARROW	`{LEFT}`
NUM LOCK	`{NUMLOCK}`
PAGE DOWN	`{PGDN}`
PAGE UP	`{PGUP}`
RETURN	`{RETURN}`
RIGHT ARROW	`{RIGHT}`
SCROLL LOCK	`{SCROLLLOCK}`
TAB	`{TAB}`
UP ARROW	`{UP}`
F1 through F15	`{F1}` through `{F15}`

In Windows, you can also specify keys combined with SHIFT and/or CTRL and/or ALT. On the Macintosh, you can also specify keys combined with SHIFT and/or CTRL and/or OPTION and/or COMMAND. To specify a key combined with another key or keys, use the following table.

To combine a key with	Precede the key code with
SHIFT	+ (plus sign)
CTRL	^ (caret)
ALT or OPTION	% (percent sign)
COMMAND	* (asterisk)

DDEExecute Method Example

This example opens a channel to Word for Windows, opens the Word document Formletr.doc, and then sends the FilePrint command to WordBasic.

```
channelNumber = Application.DDEInitiate( _
   app:="WinWord", _
   topic:="C:\WINWORD\FORMLETR.DOC")
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
```

This example opens a channel to Word for the Macintosh, opens the Word document Form Letter, and then sends the FilePrint command to WordBasic. On the Macintosh, you must use the **Shell** function to start Word, because the **DDEInitiate** method doesn't automatically start Word as it does in Windows. Also, because the **Shell** function is asynchronous, the macro may call the **DDEInitiate** method before Word has started. This example demonstrates how you can program around this by putting the **DDEInitiate** method call in a loop, testing `channelNumber` until it is no longer an error.

```
Shell "HD:Form Letter", 6
Do
   channelNumber = Application.DDEInitiate( _
      app:="MSWord", _
      topic:="HD:Form Letter")
Loop Until TypeName(channelNumber) <> "Error"
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
```

DDEInitiate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDDEInitiateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDDEInitiateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDDEInitiateA "}

Opens a DDE channel to an application.

Syntax

expression.**DDEInitiate(*App, Topic*)**

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

App Required **String**. The application name.

Topic Required **String**. Describes something in the application to which you're opening a channel − usually a document of that application.

Remarks

If successful, the **DDEInitiate** method returns the number of the open channel. All subsequent DDE functions use this number to specify the channel.

DDEInitiate Method Example

This example opens a channel to Word for Windows, opens the Word document Formletr.doc, and then sends the FilePrint command to WordBasic.

```
channelNumber = Application.DDEInitiate( _
   app:="WinWord", _
   topic:="C:\WINWORD\FORMLETR.DOC")
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
```

This example opens a channel to Word for the Macintosh, opens the Word document Form Letter, and then sends the FilePrint command to WordBasic. On the Macintosh, you must use the **Shell** function to start Word, because the **DDEInitiate** method doesn't automatically start Word as it does in Windows. Also, because the **Shell** function is asynchronous, the macro may call the **DDEInitiate** method before Word has started. This example demonstrates how you can program around this by putting the **DDEInitiate** method call in a loop, testing `channelNumber` until it is no longer an error.

```
Shell "HD:Form Letter", 6
Do
   channelNumber = Application.DDEInitiate( _
      app:="MSWord", _
      topic:="HD:Form Letter")
Loop Until TypeName(channelNumber) <> "Error"
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
```

DDEPoke Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDDEPokeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDDEPokeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDDEPokeA "}

Sends data to an application.

Syntax

expression.**DDEPoke(*Channel, Item, Data*)**

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

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

Item Required **Variant**. The item to which the data is to be sent.

Data Required **Variant**. The data to be sent to the application.

Remarks

An error occurs if the method call doesn't succeed.

DDEPoke Method Example

This example opens a channel to Word for Windows, opens the Word document Sales.doc, and then inserts the contents of cell A1 (on Sheet1) at the beginning of the document.

```
channelNumber = Application.DDEInitiate( _
   app:="WinWord", _
   topic:="C:\WINWORD\SALES.DOC")
Set rangeToPoke = Worksheets("Sheet1").Range("A1")
Application.DDEPoke channelNumber, "\StartOfDoc", rangeToPoke
Application.DDETerminate channelNumber
```

This example opens a channel to Word for the Macintosh, opens the Word document Sales Report, and then inserts the contents of cell A1 (on Sheet1) at the beginning of the document. On the Macintosh, you must use the **Shell** function to start Word, because the **DDEInitiate** method doesn't automatically start Word as it does in Windows. Also, because the **Shell** function is asynchronous, the macro may call the **DDEInitiate** method before Word has started. This example demonstrates how you can program around this by putting the **DDEInitiate** method call in a loop, testing `channelNumber` until it is no longer an error.

```
Shell "HD:Sales Report", 6
Do
   channelNumber = Application.DDEInitiate( _
      app:="WinWord", _
      topic:="HD:Sales Report")
Loop Until TypeName(channelNumber) <> "Error"
Set rangeToPoke = Worksheets("Sheet1").Range("A1")
Application.DDEPoke channelNumber, "\StartOfDoc", rangeToPoke
Application.DDETerminate channelNumber
```

DDERequest Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDDERequestC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDDERequestX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDDERequestA "}

Requests information from the specified application. This method always returns an array; for more information, see the example.

Syntax

expression.**DDERequest(*Channel, Item*)**

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

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

Item Required **String**. The item to be requested.

DDERequest Method Example

This example opens a channel to the System topic in Word for Windows and then uses the Topics item to return a list of all open documents. The list is returned in column A on Sheet1.

```
channelNumber = Application.DDEInitiate( _
   app:="WinWord", _
   topic:="System")
returnList = Application.DDERequest(channelNumber, "Topics")
For i = LBound(returnList) To UBound(returnList)
   Worksheets("Sheet1").Cells(i, 1).Formula = returnList(i)
Next i
Application.DDETerminate channelNumber
```

This example opens a channel to the System topic in Word for the Macintosh and then uses the Topics item to return a list of all open documents. The list is returned in column A on Sheet1. On the Macintosh, you must use the **Shell** function to start Word, because the **DDEInitiate** method doesn't automatically start Word as it does in Windows. Also, because the **Shell** function is asynchronous, the macro may call the **DDEInitiate** method before Word has started. This example demonstrates how you can program around this by putting the **DDEInitiate** method call in a loop, testing `channelNumber` until it is no longer an error.

```
Shell MacID("MSWD"), 6
Do
   channelNumber = Application.DDEInitiate( _
      app:="MSWord", _
      topic:="System")
Loop Until TypeName(channelNumber) <> "Error"
returnList = Application.DDERequest(channelNumber, "Topics")
For i = LBound(returnList) To UBound(returnList)
   Worksheets("Sheet1").Cells(i, 1).Formula = returnList(i)
Next i
Application.DDETerminate channelNumber
```

DDETerminate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDDETerminateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDDETerminateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDDETerminateA "}

Closes a channel to another application.

Syntax

expression.**DDETerminate(*Channel*)**

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

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

DDETerminate Method Example

This example opens a channel to Word for Windows, opens the Word document Formletr.doc, and then sends the FilePrint command to WordBasic.

```
channelNumber = Application.DDEInitiate( _
   app:="WinWord", _
   topic:="C:\WINWORD\FORMLETR.DOC")
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
```

This example opens a channel to Word for the Macintosh, opens the Word document Form Letter, and then sends the FilePrint command to WordBasic. On the Macintosh, you must use the **Shell** function to start Word, because the **DDEInitiate** method doesn't automatically start Word as it does in Windows. Also, because the **Shell** function is asynchronous, the macro may call the **DDEInitiate** method before Word has started. This example demonstrates how you can program around this by putting the **DDEInitiate** method call in a loop, testing `channelNumber` until it is no longer an error.

```
Shell "HD:Form Letter", 6
Do
   channelNumber = Application.DDEInitiate( _
      app:="MSWord", _
      topic:="HD:Form Letter")
Loop Until TypeName(channelNumber) <> "Error"
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
```

DisplayFormulaBar Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayFormulaBarC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayFormulaBarX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayFormulaBarA "}

True if the formula bar is displayed. Read/write **Boolean**.

DisplayFormulaBar Property Example

This example hides the formula bar.

```
Application.DisplayFormulaBar = False
```

DisplayStatusBar Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayStatusBarC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayStatusBarX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayStatusBarA "}

True if the status bar is displayed. Read/write **Boolean**.

DisplayStatusBar Property Example

This example saves the current state of the **DisplayStatusBar** property and then sets the property to **True** so that the status bar is visible.

```
saveStatusBar = Application.DisplayStatusBar
Application.DisplayStatusBar = True
```

DoubleClick Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDoubleClickC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDoubleClickX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDoubleClickA "}

Equivalent to double-clicking the active cell.

Syntax

expression.**DoubleClick**

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

DoubleClick Method Example

This example double-clicks the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
Application.DoubleClick
```

Evaluate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthEvaluateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthEvaluateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthEvaluateA "}

Converts a Microsoft Excel name to an object or a value.

Syntax

expression.**Evaluate(*Name*)**

expression Optional for **Application**, required for **Chart**, **DialogSheet**, and **Worksheet**. An expression that returns an object in the Applies To list.

Name Required **String**. The name of the object, using the naming convention of Microsoft Excel.

Remarks

The following types of names in Microsoft Excel can be used with this method:

- A1-style references. You can use any reference to a single cell in A1-style notation. All references are considered to be absolute references.
- Ranges. You can use the range, intersect, and union operators (colon, space, and comma, respectively) with references.
- Defined names. You can specify any name in the language of the macro.
- External references. You can use the ! operator to refer to a cell or to a name defined in another workbook – for example, `Evaluate("[BOOK1.XLS]Sheet1!A1").`

Note Using square brackets (for example, "[A1:C5]") is identical to calling the **Evaluate** method with a string argument. For example, the following expression pairs are equivalent.

```
[a1].Value = 25
Evaluate("A1").Value = 25

trigVariable = [SIN(45)]
trigVariable = Evaluate("SIN(45)")

Set firstCellInSheet = Workbooks("BOOK1.XLS").Sheets(4).[A1]
Set firstCellInSheet = Workbooks("BOOK1.XLS").Sheets(4).Evaluate("A1")
```

The advantage of using square brackets is that the code is shorter. The advantage of using **Evaluate** is that the argument is a string, so you can either construct the string in your code or use a Visual Basic variable.

Evaluate Method Example

This example turns on bold formatting in cell A1 on Sheet1.

```
Worksheets("Sheet1").Activate
boldCell = "A1"
Application.Evaluate(boldCell).Font.Bold = True
```

ExecuteExcel4Macro Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthExecuteExcel4MacroC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthExecuteExcel4MacroX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthExecuteExcel4MacroA "}

Runs a Microsoft Excel 4.0 macro function and then returns the result of the function. The return type depends on the function.

Syntax

expression.**ExecuteExcel4Macro(*String*)**

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

String Required **String**. A Microsoft Excel 4.0 macro language function without the equal sign. All references must be given as R1C1 strings. If ***String*** contains embedded double quotation marks, you must double them. For example, to run the macro function =MID("sometext",1,4), ***String*** would have to be "MID(""sometext"",1,4)".

Remarks

The Microsoft Excel 4.0 macro isn't evaluated in the context of the current workbook or sheet. This means that any references should be external and should specify an explicit workbook name. For example, to run the Microsoft Excel 4.0 macro "My_Macro" in Book1 you must use "Book1!My_Macro()". If you don't specify the workbook name, this method fails.

ExecuteExcel4Macro Method Example

This example runs the **GET.CELL(42)** macro function on cell C3 on Sheet1 and then displays the result in a message box. The **GET.CELL(42)** macro function returns the horizontal distance from the left edge of the active window to the left edge of the active cell. This macro function has no direct Visual Basic equivalent.

```
Worksheets("Sheet1").Activate
Range("C3").Select
MsgBox ExecuteExcel4Macro("GET.CELL(42)")
```

FixedDecimal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFixedDecimalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFixedDecimalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFixedDecimalA "}

All data entered after this property is set to **True** will be formatted with the number of fixed decimal places set by the **FixedDecimalPlaces** property. Read/write **Boolean**.

FixedDecimal Property Example

This example sets the **FixedDecimal** property to **True** and then sets the **FixedDecimalPlaces** property to 4. Entering "30000" after running this example produces "3" on the worksheet, and entering "12500" produces "1.25."

```
Application.FixedDecimal = True
Application.FixedDecimalPlaces = 4
```

FixedDecimalPlaces Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFixedDecimalPlacesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFixedDecimalPlacesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFixedDecimalPlacesA "}

Returns or sets the number of fixed decimal places used when the **FixedDecimal** property is set to **True**. Read/write **Long**.

FixedDecimalPlaces Property Example

This example sets the **FixedDecimal** property to **True** and then sets the **FixedDecimalPlaces** property to 4. Entering "30000" after running this example produces "3" on the worksheet, and entering "12500" produces "1.25."

```
Application.FixedDecimal = True
Application.FixedDecimalPlaces = 4
```

FullName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFullNameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFullNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFullNameA "}

Returns the name of the object, including its path on disk, as a string. 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 AddIns
   MsgBox a.FullName
Next a
```

This example displays the path and file name of the active workbook (assuming that the workbook has been saved).

```
MsgBox ActiveWorkbook.FullName
```

Help Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthHelpC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthHelpX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthHelpA "}

Displays a Help topic.

Syntax

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

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

helpFile Optional **Variant**. The name of the online Help file you want to display. If this argument isn't specified, Microsoft Excel Help is used.

helpContextID Optional **Variant**. Specifies 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
```

IgnoreRemoteRequests Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIgnoreRemoteRequestsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIgnoreRemoteRequestsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIgnoreRemoteRequestsA "}

True if remote DDE requests are ignored. Read/write **Boolean**.

IgnoreRemoteRequests Property Example

This example sets the **IgnoreRemoteRequests** property to **True** so that remote DDE requests are ignored.

```
Application.IgnoreRemoteRequests = True
```

InputBox Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthInputBoxC;vafctInputBox;vafctMsgBox "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthInputBoxX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthInputBoxA "}

Displays a dialog box for user input. Returns the information entered in the dialog box.

Syntax

expression.**InputBox(*Prompt, Title, Default, Left, Top, HelpFile, HelpContextId, Type*)**

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

Prompt Required **String**. The message to be displayed in the dialog box. This can be a string, a number, a date, or a Boolean value (Microsoft Excel automatically coerces the value to a **String** before it's displayed).

Title Optional **Variant**. The title for the input box. If this argument is omitted, the default title is "Input."

Default Optional **Variant**. Specifies a value that will appear in the text box when the dialog box is initially displayed. If this argument is omitted, the text box is left empty. This value can be a **Range** object.

Left Optional **Variant**. Specifies an x position for the dialog box in relation to the upper-left corner of the screen, in points.

Top Optional **Variant**. Specifies a y position for the dialog box in relation to the upper-left corner of the screen, in points.

HelpFile Optional **Variant**. The name of the Help file for this input box. If the ***HelpFile*** and ***HelpContextID*** arguments are present, a Help button will appear in the dialog box.

HelpContextId Optional **Variant**. The context ID number of the Help topic in ***HelpFile***.

Type Optional **Variant**. Specifies the return data type. If this argument is omitted, the dialog box returns text. Can be one or a sum of the following values.

Value	Meaning
0	A formula
1	A number
2	Text (a string)
4	A logical value (**True** or **False**)
8	A cell reference, as a **Range** object
16	An error value, such as #N/A
64	An array of values

You can use the sum of the allowable values for ***Type***. For example, for an input box that can accept both text and numbers, set ***Type*** to 1 + 2.

Remarks

Use **InputBox** to display a simple dialog box so that you can enter information to be used in a macro. The dialog box has an **OK** button and a **Cancel** button. If you choose the **OK** button, **InputBox** returns the value entered in the dialog box. If you click the **Cancel** button, **InputBox** returns **False**.

If ***Type*** is 0, **InputBox** returns the formula in the form of text – for example, "=2*PI()/360". If there are any references in the formula, they are returned as A1-style references. (Use **ConvertFormula** to convert between reference styles.)

If ***Type*** is 8, **InputBox** returns a **Range** object. You must use the **Set** statement to assign the result to a **Range** object, as shown in the following example.

```
Set myRange = Application.InputBox(prompt := "Sample", type := 8)
```

If you don't use the **Set** statement, the variable is set to the value in the range, rather than the **Range** object itself.

If you use the **InputBox** method to ask the user for a formula, you must use the **FormulaLocal** property to assign the formula to a **Range** object. The input formula will be in the user's language.

The **InputBox** method differs from the **InputBox** function in that it allows selective validation of the user's input, and it can be used with Microsoft Excel objects, error values, and formulas. Note that `Application.InputBox` calls the **InputBox** method; `InputBox` with no object qualifier calls the **InputBox** function.

InputBox Method Example

This example prompts the user for a number.

```
myNum = Application.InputBox("Enter a number")
```

This example prompts the user to select a cell on Sheet1. The example uses the ***Type*** argument to ensure that the return value is a valid cell reference (a **Range** object).

```
Worksheets("Sheet1").Activate
Set myCell = Application.InputBox( _
   prompt:="Select a cell", Type:=8)
```

Installed Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInstalledC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInstalledX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInstalledA "}

True if the add-in is installed. Read/write **Boolean**.

Remarks

Setting this property to **True** installs the add-in and calls its Auto_Add functions. Setting this property to **False** removes the add-in and calls its Auto_Remove functions.

Installed Property Example

This example uses a message box to display the installation status of the Solver add-in.

```
Set a = AddIns("Solver Add-In")
If a.Installed = True Then
   MsgBox "The Solver add-in is installed"
Else
   MsgBox "The Solver add-in is not installed"
End If
```

Interactive Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInteractiveC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInteractiveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInteractiveA "}

True if Microsoft Excel is in interactive mode; this property is usually **True**. If you set the this property to **False**, Microsoft Excel will block all input from the keyboard and mouse (except input to dialog boxes that are displayed by your code). Blocking user input will prevent the user from interfering with the macro as it moves or activates Microsoft Excel objects. Read/write **Boolean**.

Remarks

This property is useful if you're using DDE, AppleEvents, or OLE Automation to communicate with Microsoft Excel from another application.

If you set this property to **False**, don't forget to set it back to **True**. Microsoft Excel won't automatically set this property back to **True** when your macro stops running.

Interactive Property Example

This example sets the **Interactive** property to **False** while it's using DDE in Windows and then sets this property back to **True** when it's finished. This prevents the user from interfering with the macro.

```
Application.Interactive = False
Application.DisplayAlerts = False
channelNumber = Application.DDEInitiate( _
   app:="WinWord", _
   topic:="C:\WINWORD\FORMLETR.DOC")
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
Application.DisplayAlerts = True
Application.Interactive = True
```

Intersect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIntersectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIntersectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIntersectA "}

Returns a **Range** object that represents the rectangular intersection of two or more ranges.

Syntax

expression.**Intersect(*Arg1, Arg2, ...*)**

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

Arg1, Arg2, ... Required **Range**. The intersecting ranges. At least two **Range** objects must be specified.

Intersect Method Example

This example selects the intersection of two named ranges, rg1 and rg2, on Sheet1. If the ranges don't intersect, the example displays a message.

```
Worksheets("Sheet1").Activate
Set isect = Application.Intersect(Range("rg1"), Range("rg2"))
If isect Is Nothing Then
   MsgBox "Ranges do not intersect"
Else
   isect.Select
End If
```

Iteration Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIterationC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIterationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIterationA "}

True if Microsoft Excel will use iteration to resolve circular references. Read/write **Boolean**.

Iteration Property Example

This example sets the **Iteration** property to **True** so that circular references will be resolved by iteration.

```
Application.Iteration = True
```

LibraryPath Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLibraryPathC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLibraryPathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLibraryPathA "}

Returns the path to the Library folder, but without the final separator. Read-only **String**.

LibraryPath Property Example

This example opens the file Oscar.xla in the Library folder (the Macro Library folder on the Macintosh).

```
pathSep = Application.PathSeparator
f = Application.LibraryPath & pathSep & "Oscar.Xla"
Workbooks.Open filename:=f
```

MathCoprocessorAvailable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMathCoprocessorAvailableC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMathCoprocessorAvailableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMathCoprocessorAvailableA "}

True if a math coprocessor is available. Read-only **Boolean**.

MathCoprocessorAvailable Property Example

This example displays a message box if a math coprocessor isn't available.

```
If Not Application.MathCoprocessorAvailable Then
   MsgBox "This macro requires a math coprocessor"
End If
```

MaxChange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMaxChangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMaxChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMaxChangeA "}

Returns or sets the maximum amount of change between each iteration as Microsoft Excel resolves circular references. Read/write **Double**.

Remarks

The **MaxIterations** property sets the maximum number of iterations that Microsoft Excel can use when resolving circular references.

MaxChange Property Example

This example sets the maximum amound of change for each iteration to 0.1.

```
Application.MaxChange = 0.1
```

Parent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproParentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproParentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproParentA "}

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

Parent Property Example

This example displays the name of the chart that contains `myAxis`.

```
Set myAxis = Charts(1).Axes(xlValue)
MsgBox myAxis.Parent.Name
```

Run Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRunC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRunX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRunA "}

Syntax 1: Runs a macro or calls a function. This can be used to run a macro written in Visual Basic or the Microsoft Excel 4.0 macro language, or to run a function in a DLL or XLL.

Syntax 2: Runs the Microsoft Excel 4.0 macro at this location. The range must be on a macro sheet.

Syntax 1

expression.**Run(*Macro,* *Arg1, Arg2, ...*)**

Syntax 2

expression.**Run(***Arg1, Arg2, ...***)**

expression Optional for **Application**, required for **Range**. An expression that returns the application that contains the macro, or a range on a macro sheet that contains a Microsoft Excel 4.0 macro.

Macro Required **Variant** for Syntax 1 (not used with Syntax 2). The macro to run. This can be either a string with the macro name, a **Range** object indicating where the function is, or a register ID for a registered DLL (XLL) function. If a string is used, the string will be evaluated in the context of the active sheet.

Arg1, Arg2, ... Optional **Variant**. The arguments that should be passed to the function.

Remarks

You cannot use named arguments with this method. Arguments must be passed by position.

The **Run** method returns whatever the called macro returns. Objects passed as arguments to the macro are converted to values (by applying the **Value** property to the object). This means that you cannot pass objects to macros by using the **Run** method.

Run Method Example

This example shows how to call the function macro My_Func_Sum, which is defined on the macro sheet Mycustom.xlm (the macro sheet must be open). The function takes two numeric arguments (1 and 5, in this example).

```
mySum = Application.Run("MYCUSTOM.XLM!My_Func_Sum", 1, 5)
MsgBox "Macro result: " & mySum
```

TransitionMenuKey Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTransitionMenuKeyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransitionMenuKeyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransitionMenuKeyA "}

Returns or sets the alternate menu or help key, which is usually "/". Read/write **String**.

TransitionMenuKey Property Example

This example sets the transition menu key to "/" (which is the default).

```
Application.TransitionMenuKey = "/"
```

TransitionNavigKeys Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTransitionNavigKeysC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransitionNavigKeysX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransitionNavigKeysA "}

True if alternate navigation keys are active. This property is not available on the Macintosh. Read/write **Boolean**.

TransitionNavigKeys Property Example

This example displays the current state of the **Transition navigation keys** option.

```
If Application.TransitionNavigKeys Then
   keyState = "On"
Else
   keyState = "Off"
End If
MsgBox "The Transition Navigation Keys option is " & keyState
```

AddReplacement Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddReplacementC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddReplacementX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddReplacementA":1}

Adds an entry to the array of AutoCorrect replacements.

Syntax

expression.**AddReplacement(*What, Replacement*)**

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

What Required **String**. The text to be replaced. If this string already exists in the array of AutoCorrect replacements, the existing substitute text is replaced by the new text.

Replacement Required **String**. The replacement text.

AddReplacement Method Example

This example substitutes the word "Temp." for the word "Temperature" in the array of AutoCorrect replacements.

```
With Application.AutoCorrect
    .AddReplacement "Temperature", "Temp."
End With
```

AutoCorrect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoCorrectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoCorrectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoCorrectA "}

Returns an **AutoCorrect** object that represents the Microsoft Excel AutoCorrect attributes. Read-only.

AutoCorrect Property Example

This example substitutes the word "Temp." for the word "Temperature" in the array of AutoCorrect replacements.

```
With Application.AutoCorrect
   .AddReplacement "Temperature", "Temp."
End With
```

CapitalizeNamesOfDays Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCapitalizeNamesOfDaysC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCapitalizeNamesOfDaysX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCapitalizeNamesOfDaysA":1}

True if the first letter of day names is capitalized automatically. Read/write **Boolean**.

CapitalizeNamesOfDays Property Example

This example sets Microsoft Excel to capitalize the first letter of the names of days.

```
With Application.AutoCorrect
    .CapitalizeNamesOfDays = True
    .ReplaceText = True
End With
```

DeleteReplacement Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDeleteReplacementC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDeleteReplacementX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDeleteReplacementA":1}

Deletes an entry from the array of AutoCorrect replacements.

Syntax

expression.**DeleteReplacement(*What*)**

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

What Required **String**. The text to be replaced, as it appears in the row to be deleted from the array of AutoCorrect replacements. If this string doesn't exist in the array of AutoCorrect replacements, this method fails.

DeleteReplacement Method Example

This example removes the word "Temperature" from the array of AutoCorrect replacements.

```
With Application.AutoCorrect
   .DeleteReplacement "Temperature"
End With
```

ReplaceText Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReplaceTextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReplaceTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReplaceTextA":1}

True if text in the list of AutoCorrect replacements is replaced automatically. Read/write **Boolean**.

ReplaceText Property Example

This example turns off automatic text replacement.

```
With Application.AutoCorrect
    .CapitalizeNamesOfDays = True
    .ReplaceText = False
End With
```

TwoInitialCapitals Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTwoInitialCapitalsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTwoInitialCapitalsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTwoInitialCapitalsA":1}

True if words that begin with two capital letters are corrected automatically. Read/write **Boolean**.

TwoInitialCapitals Property Example

This example sets Microsoft Excel to correct words that begin with two capital letters.

```
With Application.AutoCorrect
    .TwoInitialCapitals = True
    .ReplaceText = True
End With
```

AskToUpdateLinks Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAskToUpdateLinksC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAskToUpdateLinksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAskToUpdateLinksA "}

True if Microsoft Excel asks the user to update links when opening files with links. **False** if links are automatically updated with no dialog box. Read/write **Boolean**.

AskToUpdateLinks Property Example

This example sets Microsoft Excel to ask the user to update links whenever a file that contains links is opened.

```
Application.AskToUpdateLinks = True
```

AutoUpdate Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoUpdateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoUpdateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoUpdateA "}

True if the OLE object is updated automatically when the source changes. Valid only if the object is linked (its **OLEType** property must be **xlOLELink**). Read-only **Boolean**.

AutoUpdate Property Example

This example displays the status of automatic updating for all OLE objects on Sheet1.

```
Worksheets("Sheet1").Activate
Range("A1").Value = "Name"
Range("B1").Value = "Link Status"
Range("C1").Value = "AutoUpdate Status"
i = 2
For Each obj In ActiveSheet.OLEObjects
   Cells(i, 1) = obj.Name
   If obj.OLEType = xlOLELink Then
      Cells(i, 2) = "Linked"
      Cells(i, 3) = obj.AutoUpdate
   Else
      Cells(i, 2) = "Embedded"
   End If
   i = i + 1
Next
```

Backward Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBackwardC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBackwardX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBackwardA "}

Returns or sets the number of periods (or units on a scatter chart) that the trendline extends backward. Read/write **Long**

Backward Property Example

This example sets the number of units that the trendline on Chart1 extends forward and backward. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
With Charts("Chart1").SeriesCollection(1).Trendlines(1)
    .Forward = 5
    .Backward = .5
End With
```

BlackAndWhite Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBlackAndWhiteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBlackAndWhiteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBlackAndWhiteA "}

True if elements of the document will be printed in black and white. Read/write **Boolean**.

Remarks

This property applies only to worksheet pages.

BlackAndWhite Property Example

This example causes Sheet1 to be printed in black and white.

```
Worksheets("Sheet1").PageSetup.BlackAndWhite = True
```

Category Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCategoryC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCategoryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCategoryA "}

Returns or sets the category for the specified name in the language of the macro. The name must refer to a custom function or command. Read/write **String**.

Category Property Example

This example assumes that you created a custom function or command on a Microsoft Excel 4.0 macro sheet. The example displays the function category in the language of the macro. It assumes that the name of the custom function or command is the only name in the workbook.

```
With ActiveWorkbook.Names(1)
   If .MacroType <> xlNone Then
      MsgBox "The category for this name is " & .Category
   Else
      MsgBox "This name does not refer to" & _
         " a custom function or command."
   End If
End With
```

ChangeFileAccess Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthChangeFileAccessC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthChangeFileAccessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthChangeFileAccessA "}

Changes the access permissions for the workbook. This may require an updated version to be loaded from the disk.

Syntax

expression.**ChangeFileAccess(*Mode, WritePassword, Notify*)**

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

Mode Optional **Variant**. Specifies the new access mode. Can be one of the following **XlFileAccess** constants: **xlReadWrite** or **xlReadOnly**.

WritePassword Optional **Variant**. Specifies the write-reserved password if the file is write reserved and ***Mode*** is **xlReadWrite**. Ignored if there's no password for the file or if ***Mode*** is **xlReadOnly**.

Notify Optional **Variant**. **True** (or omitted) to notify the user if the file cannot be immediately accessed.

Remarks

If you have a file open in read-only mode, you don't have exclusive access to the file. If you change a file from read-only to read/write, Microsoft Excel must load a new copy of the file to ensure that no changes were made while you had the file open as read-only.

ChangeFileAccess Method Example

This example sets the active workbook to read-only.

```
ActiveWorkbook.ChangeFileAccess Mode:=xlReadOnly
```

ChartGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthChartGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthChartGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthChartGroupsA "}

Returns an object that represents either a single chart group (a **ChartGroup** object, Syntax 1) or a collection of all the chart groups in the chart (a **ChartGroups** object, Syntax 2). The returned collection includes every type of group.

Syntax 1

expression.**ChartGroups(*Index*)**

Syntax 2

expression.**ChartGroups**

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

Index Optional **Variant**. The chart group number.

ChartGroups Method Example

This example turns on up and down bars for chart group one on Chart1 and then sets their colors. The example should be run on a 2-D line chart containing two series that intersect at one or more data points.

```
With Charts("Chart1").ChartGroups(1)
    .HasUpDownBars = True
    .DownBars.Interior.ColorIndex = 3
    .UpBars.Interior.ColorIndex = 5
End With
```

ClearArrows Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearArrowsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearArrowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearArrowsA "}

Clears the tracer arrows from the worksheet. Tracer arrows are added by using the auditing feature.

Syntax

expression.**ClearArrows**

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

ClearArrows Method Example

This example clears tracer arrows from Sheet1.

```
Worksheets("Sheet1").ClearArrows
```

Copy Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCopyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCopyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCopyA "}

Syntax 1: Copies the object to the Clipboard. Copies a picture of the point or series to the Clipboard.

Syntax 2: Copies the **Range** to the specified range or to the Clipboard.

Syntax 3: Copies the sheet to another location in the workbook.

Syntax 1

expression.**Copy**

Syntax 2

expression.**Copy(*Destination*)**

Syntax 3

expression.**Copy(*Before, After*)**

expression Required. An expression that returns an object in the Applies To list. To copy an entire chart sheet, use Syntax 3 with the **Chart** object. To copy only the chart area, use Syntax 1 with the **ChartArea** object.

Destination Optional **Variant**. Specifies the new range to which the specified range will be copied. If this argument is omitted, Microsoft Excel copies the range to the Clipboard.

Before Syntax 3: Optional **Variant**. The sheet before which the copied sheet will be placed. You cannot specify ***Before*** if you specify ***After***.

After Optional **Variant**. The sheet after which the copied sheet will be placed. You cannot specify ***After*** if you specify ***Before***.

Remarks

If you don't specify either ***Before*** or ***After***, Microsoft Excel creates a new workbook that contains the copied sheet.

Copy Method Example

This example copies Sheet1, placing the copy after Sheet3.

```
Worksheets("Sheet1").Copy after := Worksheets("Sheet3")
```

This example copies the used range on Sheet1, creates a new worksheet, and then pastes the values of the copied range onto the new worksheet.

```
Worksheets("Sheet1").UsedRange.Copy
Set newSheet = Worksheets.Add
newSheet.Range("A1").PasteSpecial Paste:=xlValues
```

This example copies the formulas in cells A1:D4 on Sheet1 into cells E5:H8 on Sheet2.

```
Worksheets("Sheet1").Range("A1:D4").Copy _
   destination:=Worksheets("Sheet2").Range("E5")
```

MaxIterations Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMaxIterationsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMaxIterationsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMaxIterationsA "}

Returns or sets the maximum number of iterations that Microsoft Excel can use to resolve a circular reference. Read/write **Long**.

Remarks

The **MaxChange** property sets the maximum amount of change between each iteration when Microsoft Excel is resolving circular references.

MaxIterations Property Example

This example sets the maximum number of iterations at 1000.

```
Application.MaxIterations = 1000
```

MemoryFree Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMemoryFreeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMemoryFreeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMemoryFreeA "}

Returns the amount of memory that's still available for Microsoft Excel to use, in bytes. Read-only **Long**.

MemoryFree Property Example

This example displays a message box showing the number of free bytes.

```
MsgBox "Microsoft Excel has " & Application.MemoryFree & " bytes free"
```

MemoryTotal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMemoryTotalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMemoryTotalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMemoryTotalA "}

Returns the total amount of memory (in bytes) that's available to Microsoft Excel, including memory already in use. Read-only **Long**.

Remarks

MemoryTotal is equal to **MemoryUsed** + **MemoryFree.**

MemoryTotal Property Example

This example displays a message box showing the total number of available bytes.

```
MsgBox "Microsoft Excel has " & Application.MemoryTotal & _
   " total bytes available"
```

MouseAvailable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMouseAvailableC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMouseAvailableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMouseAvailableA "}

True if a mouse is available (always **True** on the Macintosh). Read-only **Boolean**.

MouseAvailable Property Example

This example displays a message if a mouse isn't available.

```
If Application.MouseAvailable = False Then
   MsgBox "Your system does not have a mouse"
End If
```

Move Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthMoveC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthMoveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthMoveA "}

Moves the sheet to another location in the workbook.

Syntax

expression.**Move(*Before, After*)**

expression Required. An expression that returns an object in the Applies To list.

Before Optional **Variant**. The sheet before which the moved sheet will be placed. You cannot specify ***Before*** if you specify ***After***.

After Optional **Variant**. The sheet after which the moved sheet will be placed. You cannot specify ***After*** if you specify ***Before***.

Remarks

If you don't specify either ***Before*** or ***After***, Microsoft Excel creates a new workbook that contains the moved sheet.

Move Method Example

This example moves Sheet1 after Sheet3 in the active workbook.

```
Worksheets("Sheet1").Move _
    after:=Worksheets("Sheet3")
```

MoveAfterReturn Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMoveAfterReturnC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMoveAfterReturnX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMoveAfterReturnA "}

True if the active cell will be moved as soon as the ENTER (RETURN) key is pressed. Read/write **Boolean**.

Remarks

Use the **MoveAfterReturnDirection** property to specify the direction in which the active cell is to be moved.

MoveAfterReturn Property Example

This example sets the **MoveAfterReturn** property to **True**.

```
Application.MoveAfterReturn = True
```

OperatingSystem Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOperatingSystemC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOperatingSystemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOperatingSystemA "}

Returns the name and version number of the current operating system – for example, "Windows (32-bit) 4.00" or "Macintosh 7.00". Read-only **String**.

OperatingSystem Property Example

This example displays the name of the operating system.

```
MsgBox "Microsoft Excel is using " & Application.OperatingSystem
```

OrganizationName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOrganizationNameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOrganizationNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOrganizationNameA "}

Returns the registered organization name. Read-only **String**.

OrganizationName Property Example

This example displays the registered organization name.

```
MsgBox "The registered organization is " & Application.OrganizationName
```

Path Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPathC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPathA "}

Returns the complete path of the object, excluding the final separator and name of the object. Read-only **String**.

Remarks

Using this property without an object qualifier is equivalent to `Application.Path` (this returns the path to the Microsoft Excel application).

Path Property Example

This example displays the complete path to Microsoft Excel.

```
MsgBox "The path is " & Application.Path
```

PathSeparator Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPathSeparatorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPathSeparatorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPathSeparatorA "}

Returns the path separator character ("\" in Windows, or ":" on the Macintosh). Read-only **String**.

PathSeparator Property Example

This example displays the current path separator.

```
MsgBox "The path separator character is " & Application.PathSeparator
```

PrintOut Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPrintOutC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPrintOutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPrintOutA "}

Prints the object.

Syntax

expression.**PrintOut(*from, To, Copies, Preview, ActivePrinter, PrintToFile, Collate*)**

expression Required. An expression that returns an object in the Applies To list.

From Optional **Variant**. The number of the page at which to start printing. If this argument is omitted, printing starts at the beginning.

To Optional **Variant**. The number of the last page to print. If this argument is omitted, printing ends with the last page.

Copies Optional **Variant**. The number of copies to print. If this argument is omitted, one copy is printed.

Preview Optional **Variant**. **True** to have Microsoft Excel invoke print preview before printing the object. **False** (or omitted) to print the object immediately.

ActivePrinter Optional **Variant**. Sets the name of the active printer.

PrintToFile Optional **Variant**. **True** to print to a file. Microsoft Excel prompts the user to enter the name of the output file. There's no way to specify the name of the output file from Visual Basic.

Collate Optional **Variant**. **True** to collate multiple copies.

Remarks

"Pages" in the descriptions of ***From*** and ***To*** refers to printed pages – not overall pages in the sheet or workbook.

This method applies to the **Window** object only when it's the Info window.

PrintOut Method Example

This example prints the active sheet.

```
ActiveSheet.PrintOut
```

PrintPreview Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPrintPreviewC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPrintPreviewX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPrintPreviewA "}

Shows a preview of the object as it would look when printed.

Syntax

expression.**PrintPreview**

expression Required. An expression that returns an object in the Applies To list.

PrintPreview Method Example

This example displays Sheet1 in print preview.

```
Worksheets("Sheet1").PrintPreview
```

RecordRelative Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRecordRelativeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRecordRelativeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRecordRelativeA "}

True if macros are recorded using relative references; **False** if recording is absolute. Read-only **Boolean**.

RecordRelative Property Example

This example displays the address of the active cell on Sheet1 in A1 style if **RecordRelative** is **False**; otherwise, it displays the address in R1C1 style.

```
Worksheets("Sheet1").Activate
If Application.RecordRelative = False Then
   MsgBox ActiveCell.Address(ReferenceStyle:=xlA1)
Else
   MsgBox ActiveCell.Address(ReferenceStyle:=xlR1C1)
End If
```

ReferenceStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReferenceStyleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReferenceStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReferenceStyleA "}

Returns or sets how Microsoft Excel displays cell references and row and column headings in either A1 or R1C1 reference style. Can be one of the following **XlReferenceStyle** constants: **xlA1** or **xlR1C1**. Read/write **Long**.

ReferenceStyle Property Example

This example displays the current reference style.

```
If Application.ReferenceStyle = xlR1C1 Then
   MsgBox ("Microsoft Excel is using R1C1 references")
Else
   MsgBox ("Microsoft Excel is using A1 references")
End If
```

Repeat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRepeatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRepeatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRepeatA "}

Repeats the last user-interface action.

Syntax

expression.**Repeat**

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

Remarks

This method repeats only the last action taken by the user before running the macro, and it must be the first line in the macro. It cannot be used to repeat Visual Basic commands.

Repeat Method Example

This example repeats the last user-interface command. The example must be the first line in a macro.

```
Application.Repeat
```

Save Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSaveC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSaveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSaveA "}

Saves changes to the specified workbook.

Syntax

expression.**Save**

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

Remarks

To open a workbook file, use the **Open** method.

To mark a workbook as saved without writing it to a disk, set its **Saved** property to **True**.

The first time you save a workbook, use the **SaveAs** method to specify a name for the file.

Save Method Example

This example saves the active workbook.

```
ActiveWorkbook.Save
```

This example saves all open workbooks and then closes Microsoft Excel.

```
For Each w In Application.Workbooks
   w.Save
Next w
Application.Quit
```

Scenarios Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthScenariosC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthScenariosX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthScenariosA "}

Returns an object that represents either a single scenario (a **Scenario** object, Syntax 1) or a collection of scenarios (a **Scenarios** object, Syntax 2) on the worksheet.

Syntax 1

expression.**Scenarios(*Index*)**

Syntax 2

expression.**Scenarios**

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

Index Optional **Variant**. The name or number of the scenario. Use an array to specify more than one scenario.

Scenarios Method Example

This example sets the comment for the first scenario on Sheet1.

```
Worksheets("Sheet1").Scenarios(1).Comment = _
    "Worst-case July 1993 sales"
```

ScreenUpdating Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproScreenUpdatingC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproScreenUpdatingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproScreenUpdatingA "}

True if screen updating is turned on. Read/write **Boolean**.

Remarks

Turn screen updating off to speed up your macro code. You won't be able to see what the macro is doing, but it will run faster.

Remember to set the **ScreenUpdating** property back to **True** when your macro ends (older versions of Microsoft Excel automatically reset this property, but Microsoft Excel 97 does not).

ScreenUpdating Property Example

This example demonstrates how turning off screen updating can make your code run faster. The example hides every other column on Sheet1, while keeping track of the time it takes to do so. The first time the example hides the columns, screen updating is turned on; the second time, screen updating is turned off. When you run this example, you can compare the respective running times, which are displayed in the message box.

```
Dim elapsedTime(2)
Application.ScreenUpdating = True
For i = 1 To 2
   If i = 2 Then Application.ScreenUpdating = False
   startTime = Time
   Worksheets("Sheet1").Activate
   For Each c In ActiveSheet.Columns
      If c.Column Mod 2 = 0 Then
         c.Hidden = True
      End If
   Next c
   stopTime = Time
   elapsedTime(i) = (stopTime - startTime) * 24 * 60 * 60
Next i
Application.ScreenUpdating = True
MsgBox "Elapsed time, screen updating on: " & elapsedTime(1) & _
      " sec." & Chr(13) & _
      "Elapsed time, screen updating off: " & elapsedTime(2) & _
      " sec."
```

SendKeys Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSendKeysC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSendKeysX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSendKeysA "}

Sends keystrokes to the active application. This method isn't available on the Macintosh.

Syntax

expression.**SendKeys(*Keys, Wait*)**

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

Keys Required **Variant**. The key or key combination you want to send to the application, as text.

Wait Optional **Variant**. **True** to have Microsoft Excel wait for the keys to be processed before returning control to the macro. **False** (or omitted) to continue running the macro without waiting for the keys to be processed.

Remarks

This method places keystrokes in a key buffer. In some cases, you must call this method before you call the method that will use the keystrokes. For example, to send a password to a dialog box, you must call the **SendKeys** method before you display the dialog box.

The ***Keys*** argument can specify any single key or any key combined with ALT, CTRL, or SHIFT (or any combination of those keys). Each key is represented by one or more characters, such as `"a"` for the character a, or `"{ENTER}"` for the ENTER key.

To specify characters that aren't displayed when you press the corresponding key (for example, ENTER or TAB), use the codes listed in the following table. Each code in the table represents one key on the keyboard.

Key	Code
BACKSPACE	`{BACKSPACE}` or `{BS}`
BREAK	`{BREAK}`
CAPS LOCK	`{CAPSLOCK}`
CLEAR	`{CLEAR}`
DELETE or DEL	`{DELETE}` or `{DEL}`
DOWN ARROW	`{DOWN}`
END	`{END}`
ENTER (numeric keypad)	`{ENTER}`
ENTER	~ (tilde)
ESC	`{ESCAPE}` or `{ESC}`
HELP	`{HELP}`
HOME	`{HOME}`
INS	`{INSERT}`
LEFT ARROW	`{LEFT}`
NUM LOCK	`{NUMLOCK}`
PAGE DOWN	`{PGDN}`
PAGE UP	`{PGUP}`
RETURN	`{RETURN}`
RIGHT ARROW	`{RIGHT}`
SCROLL LOCK	`{SCROLLLOCK}`
TAB	`{TAB}`

UP ARROW	`{UP}`
F1 through F15	`{F1}` through `{F15}`

You can also specify keys combined with SHIFT and/or CTRL and/or ALT. To specify a key combined with another key or keys, use the following table.

To combine a key with	Precede the key code with
SHIFT	+ (plus sign)
CTRL	^ (caret)
ALT	% (percent sign)

SendKeys Method Example

This example uses the **SendKeys** method to quit Microsoft Excel for Windows.

```
Application.SendKeys("%fx")
```

StartupPath Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStartupPathC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStartupPathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStartupPathA "}

Returns the complete path of the startup folder, excluding the final separator. Read-only **String**.

StartupPath Property Example

This example displays the full path to the Microsoft Excel startup folder.

```
MsgBox Application.StartupPath
```

StatusBar Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStatusBarC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStatusBarX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStatusBarA "}

Returns or sets the text in the status bar. Read/write **String**.

Remarks

This property returns **False** if Microsoft Excel has control of the status bar. To restore the default status bar text, set the property to **False**; this works even if the status bar is hidden.

StatusBar Property Example

This example sets the status bar text to "Please be patient..." before it opens the workbook Large.xls, and then it restores the default text.

```
oldStatusBar = Application.DisplayStatusBar
Application.DisplayStatusBar = True
Application.StatusBar = "Please be patient..."
Workbooks.Open filename:="LARGE.XLS"
Application.StatusBar = False
Application.DisplayStatusBar = oldStatusBar
```

ThisWorkbook Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproThisWorkbookC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproThisWorkbookX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproThisWorkbookA "}

Returns a **Workbook** object that represents the workbook where the current macro code is running. Read-only.

Remarks

Use this property to refer to the workbook that contains your macro code. **ThisWorkbook** is the only way to refer to an add-in workbook from inside the add-in itself. The **ActiveWorkbook** property doesn't return the add-in workbook; it returns the workbook that's *calling* the add-in.The **Workbooks** property may fail, as the workbook name probably changed when you created the add-in. **ThisWorkbook** always returns the workbook in which the code is running.

For example, use code such as the following to activate a dialog sheet stored in your add-in workbook.

```
ThisWorkbook.DialogSheets(1).Show
```

This property can be used only from inside Microsoft Excel. You cannot use it to access a workbook from any other application.

ThisWorkbook Property Example

This example closes the workbook that contains the example code. Changes to the workbook, if any, aren't saved.

```
ThisWorkbook.Close SaveChanges:=False
```

Undo Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUndoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUndoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUndoA "}

Cancels the last user-interface action.

Syntax

expression.**Undo**

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

Remarks

This method undoes only the last action taken by the user before running the macro, and it must be the first line in the macro. It cannot be used to undo Visual Basic commands.

Undo Method Example

This example cancels the last user-interface action. The example must be the first line in a macro.

```
Application.Undo
```

Union Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUnionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUnionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUnionA "}

Returns the union of two or more ranges.

Syntax

expression.**Union(*Arg1, Arg2, ...*)**

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

Arg1, Arg2, ... Required **Range**. At least two **Range** objects must be specified.

Union Method Example

This example fills the union of two named ranges, Range1 and Range2, with the formula =RAND().

```
Worksheets("Sheet1").Activate
Set bigRange = Application.Union(Range("Range1"), Range("Range2"))
bigRange.Formula = "=RAND()"
```

Unprotect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUnprotectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUnprotectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUnprotectA "}

Removes protection from a sheet or workbook. This method has no effect if the sheet or workbook isn't protected.

Syntax

expression.**Unprotect(*Password*)**

expression Required. An expression that returns a **Chart**, **Workbook**, or **Worksheet** object.

Password Optional **Variant**. A string that denotes the case-sensitive password to use to unprotect the sheet or workbook. If the sheet or workbook isn't protected with a password, this argument is ignored. If you omit this argument for a sheet that's protected with a password, you'll be prompted for the password. If you omit this argument for a workbook that's protected with a password, the method fails.

Remarks

If you forget the password, you cannot unprotect the sheet or workbook. It's a good idea to keep a list of your passwords and their corresponding document names in a safe place.

Unprotect Method Example

This example removes protection from the active workbook.

```
ActiveWorkbook.Unprotect
```

UsableHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUsableHeightC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUsableHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUsableHeightA "}

Returns the maximum height of the space that a window can occupy in the application window area, in points. Read-only **Double**.

UsableHeight Property Example

This example expands the active window to the maximum size available (assuming that the window isn't already maximized).

```
With ActiveWindow
    .WindowState = xlNormal
    .Top = 1
    .Left = 1
    .Height = Application.UsableHeight
    .Width = Application.UsableWidth
End With
```

UsableWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUsableWidthC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUsableWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUsableWidthA "}

Returns the maximum width of the space that a window can occupy in the application window area, in points. Read-only **Double**.

UsableWidth Property Example

This example expands the active window to the maximum size available (assuming that the window isn't already maximized).

```
With ActiveWindow
    .WindowState = xlNormal
    .Top = 1
    .Left = 1
    .Height = Application.UsableHeight
    .Width = Application.UsableWidth
End With
```

UserName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUserNameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUserNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUserNameA "}

Returns or sets the name of the current user. Read/write **String**.

UserName Property Example

This example displays the name of the current user.

```
MsgBox "Current user is " & Application.UserName
```

Version Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVersionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVersionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVersionA "}

Returns the Microsoft Excel version number. Read-only **String**.

Version Property Example

This example displays a message box that contains the Microsoft Excel version number and the name of the operating system.

```
MsgBox "Welcome to Microsoft Excel version " & _
   Application.Version & " running on " & _
   Application.OperatingSystem & "!"
```

Wait Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthWaitC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthWaitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthWaitA "}

Pauses a running macro until a specified time.

Important The **Wait** method suspends all Microsoft Excel activity and may prevent you from performing other operations on your computer while **Wait** is in effect. However, background processes such as printing and recalculation continue.

Syntax

expression.**Wait(*Time*)**

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

Time Required **Variant**. The time at which you want the macro to resume, in Microsoft Excel date format.

Wait Method Example

This example pauses a running macro until 6:23 P.M. today.

```
Application.Wait "18:23:00"
```

This example pauses a running macro for approximately 10 seconds.

```
newHour = Hour(Now())
newMinute = Minute(Now())
newSecond = Second(Now()) + 10
waitTime = TimeSerial(newHour, newMinute, newSecond)
Application.Wait waitTime
```

WindowsForPens Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWindowsForPensC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWindowsForPensX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWindowsForPensA "}

True if the computer is running under Microsoft Windows for Pen Computing. Read-only **Boolean**.

WindowsForPens Property Example

This example shows how to limit handwriting recognition to numbers and punctuation only if Microsoft Windows for Pen Computing is running.

```
If Application.WindowsForPens Then
   Application.ConstrainNumeric = True
End If
```

WindowState Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWindowStateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWindowStateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWindowStateA "}

Returns or sets the state of the window. Can be one of the following **XlWindowState** constants: **xlMaximized**, **xlMinimized**, or **xlNormal**. Read/write **Long**.

WindowState Property Example

This example maximizes the application window in Microsoft Excel for Windows. (This property cannot be set on the Macintosh.)

```
Application.WindowState = xlMaximized
```

This example expands the active window to the maximum size available (assuming that the window isn't already maximized).

```
With ActiveWindow
    .WindowState = xlNormal
    .Top = 1
    .Left = 1
    .Height = Application.UsableHeight
    .Width = Application.UsableWidth
End With
```

Area3DGroup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproArea3DGroupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproArea3DGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproArea3DGroupA "}

Returns a **ChartGroup** object that represents the area chart group on a 3-D chart. Read-only.

Area3DGroup Property Example

This example turns on drop lines for the 3-D area chart group.

```
Charts(1).Area3DGroup.HasDropLines = True
```

AreaGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAreaGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAreaGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAreaGroupsA "}

On a 2-D chart, returns an object that represents either a single area chart group (a **ChartGroup** object, Syntax 1) or a collection of the area chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**AreaGroups(*Index*)**

Syntax 2

expression.**AreaGroups**

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

Index Optional **Variant**. The chart group number.

AreaGroups Method Example

This example turns on drop lines for the 2-D area chart group.

```
Charts(1).AreaGroups(1).HasDropLines = True
```

AutoScaling Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoScalingC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoScalingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoScalingA "}

True if Microsoft Excel scales a 3-D chart so that it's closer in size to the equivalent 2-D chart. The **RightAngleAxes** property must be **True**. Read/write **Boolean**.

AutoScaling Property Example

This example automatically scales Chart1. The example should be run on a 3-D chart.

```
With Charts("Chart1")
    .RightAngleAxes = True
    .AutoScaling = True
End With
```

Axes Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAxesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAxesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAxesA "}

Returns an object that represents either a single axis (an **Axis** object, Syntax 1) or a collection of the axes on the chart (an **Axes** collection, Syntax 2).

Syntax 1

expression.**Axes(*Type, AxisGroup*)**

Syntax 2

expression.**Axes**

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

Type Optional **Variant**. Specifies the axis to return. Can be one of the following **XlAxisType** constants: **xlValue**, **xlCategory**, or **xlSeriesAxis** (**xlSeriesAxis** is valid only for 3-D charts).

AxisGroup Optional **Variant**. Specifies the axis group. Can be one of the following **XlAxisGroup** constants: **xlPrimary** or **xlSecondary**. If this argument is omitted, the primary group is used. 3-D charts have only one axis group.

Axes Method Example

This example adds an axis label to the category axis in Chart1.

```
With Charts("Chart1").Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Text = "July Sales"
End With
```

This example turns off major gridlines for the category axis in Chart1.

```
Charts("Chart1").Axes(xlCategory).HasMajorGridlines = False
```

This example turns off all gridlines for all axes in Chart1.

```
For Each a In Charts("Chart1").Axes
    a.HasMajorGridlines = False
    a.HasMinorGridlines = False
Next a
```

AxisBetweenCategories Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAxisBetweenCategoriesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAxisBetweenCategoriesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAxisBetweenCategoriesA "}

True if the value axis crosses the category axis between categories. Read/write **Boolean**.

Remarks

This property applies only to category axes, and it doesn't apply to 3-D charts.

AxisBetweenCategories Property Example

This example causes the value axis in Chart1 to cross the category axis between categories.

```
Charts("Chart1").Axes(xlCategory).AxisBetweenCategories = True
```

AxisGroup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAxisGroupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAxisGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAxisGroupA "}

Returns the group for the specified axis, chart group, or series. Can be one of the following **XlAxisGroup** constants: **xlPrimary** or **xlSecondary**. Read/write **Long** for **Series**; read-only **Long** for **Axis** and **ChartGroup**.

Remarks

For 3-D charts, only **xlPrimary** is valid.

AxisGroup Property Example

This example deletes the value axis in Chart1 if the axis is in the secondary group.

```
With Charts("Chart1").Axes(xlValue)
   If .AxisGroup = xlSecondary Then .Delete
End With
```

Bar3DGroup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBar3DGroupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBar3DGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBar3DGroupA "}

Returns a **ChartGroup** object that represents the bar chart group on a 3-D chart. Read-only.

Bar3DGroup Property Example

This example sets the space between bar clusters in the 3-D bar chart group to be 50 percent of the bar width.

```
Charts(1).BarGroup3DGroup.GapWidth = 50
```

BarGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthBarGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthBarGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthBarGroupsA "}

On a 2-D chart, returns an object that represents either a single bar chart group (a **ChartGroup** object, Syntax 1) or a collection of the bar chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**BarGroups(*Index*)**

Syntax 2

expression.**BarGroups**

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

Index Optional **Variant**. Specifies the chart group.

BarGroups Method Example

This example sets the space between bar clusters in the 2-D bar chart group to be 50 percent of the bar width.

```
Charts(1).BarGroups(1).GapWidth = 50
```

ChartArea Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChartAreaC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChartAreaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChartAreaA "}

Returns a **ChartArea** object that represents the complete chart area for the chart. Read-only.

ChartArea Property Example

This example sets the chart area interior color of Chart1 to red and sets the border color to blue.

```
With Charts("Chart1").ChartArea
    .Interior.ColorIndex = 3
    .Border.ColorIndex = 5
End With
```

Clear Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearA "}

ChartArea, **Legend**, or **Range** object: Clears the entire object.

ActiveX list box or combo box: Removes all entries from the list.

Syntax

expression.**Clear**

expression Required. An expression that returns an object in the Applies To list..

Clear Method Example

This example clears the formulas and formatting in cells A1:G37 on Sheet1.

```
Worksheets("Sheet1").Range("A1:G37").Clear
```

This example clears the chart area (the chart data and formatting) of Chart1.

```
Charts("Chart1").ChartArea.Clear
```

ClearContents Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearContentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearContentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearContentsA "}

Clears the formulas from the range. Clears the data from a chart but leaves the formatting.

Syntax

expression.**ClearContents**

expression Required. An expression that returns a **Chart** or **Range** object.

ClearContents Method Example

This example clears the formulas from cells A1:G37 on Sheet1 but leaves the formatting intact.

```
Worksheets("Sheet1").Range("A1:G37").ClearContents
```

This example clears the chart data from Chart1 but leaves the formatting intact.

```
Charts("Chart1").ChartArea.ClearContents
```

Column3DGroup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColumn3DGroupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColumn3DGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColumn3DGroupA "}

Returns a **ChartGroup** object that represents the column chart group on a 3-D chart. Read-only.

Column3DGroup Property Example

This example sets the space between column clusters in the 3-D column chart group to be 50 percent of the column width.

```
Charts(1).Column3DGroup.GapWidth = 50
```

ColumnGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthColumnGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthColumnGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthColumnGroupsA "}

On a 2-D chart, returns an object that represents either a single column chart group (a **ChartGroup** object, Syntax 1) or a collection of the column chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**ColumnGroups(*Index*)**

Syntax 2

expression.**ColumnGroups**

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

Index Optional **Variant**. Specifies the chart group.

ColumnGroups Method Example

This example sets the space between column clusters in the 2-D column chart group to be 50 percent of the column width.

```
Charts(1).ColumnGroups(1).GapWidth = 50
```

Corners Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCornersC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCornersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCornersA "}

Returns a **Corners** object that represents the corners of a 3-D chart. Read-only.

Corners Property Example

This example selects the corners of Chart1. The example should be run on a 3-D chart (the **Select** method fails on any other chart type).

```
With Charts("Chart1")
    .Activate
    .Corners.Select
End With
```

CreatePublisher Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCreatePublisherC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCreatePublisherX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCreatePublisherA "}

Creates a publisher based on a **Chart** or a **Range** object. Available only on the Macintosh with System 7 or later.

Syntax

expression.**CreatePublisher(*Edition, Appearance, Size, ContainsPICT, ContainsBIFF, ContainsRTF, ContainsVALU*)**

expression Required. An expression that returns a **Chart** or **Range** object.

Edition Optional **Variant**. The filename of the edition to be created. If this argument is omitted, "<Document Name> Edition #*n*" is used.

Appearance Optional **Variant**. One of **xlPrinter** or **xlScreen**.

Size Optional **Variant** (used only with **Chart** objects). One of **xlPrinter** or **xlScreen**.

ContainsPICT Optional **Variant**. **True** to include PICT format in the publisher. The default value is **True**.

ContainsBIFF Optional **Variant**. **True** to include BIFF format in the publisher. The default value for **Range** is **True**; the default value for **Chart** is **False**.

ContainsRTF Optional **Variant**. **True** to include RTF format in the publisher. The default value for **Range** is **True**; the default value for **Chart** is **False**.

ContainsVALU Optional **Variant**. **True** to include VALU format in the publisher. The default value for **Range** is **True**; the default value for **Chart** is **False**.

CreatePublisher Method Example

This example creates a publisher based on cells A1:A20 on Sheet1.

```
Worksheets("Sheet1").Range("A1:A20").CreatePublisher "stock data"
```

Crosses Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCrossesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCrossesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCrossesA "}

Returns or sets the point on the specified axis where the other axis crosses. Read/write **Long**.

Can be one of the following **XlAxisCrosses** constants.

Constant	Meaning
xlAxisCrossesAutomatic	Microsoft Excel sets the axis crossing point.
xlMinimum	The axis crosses at the minimum value.
xlMaximum	The axis crosses at the maximum value.
xlAxisCrossesCustom	The **CrossesAt** property specifies the axis crossing point.

Remarks

This property isn't available for 3-D charts or radar charts.

This property can be used for both category and value axes. On the category axis, **xlMinimum** sets the value axis to cross at the first category, and **xlMaximum** sets the value axis to cross at the last category.

Note that **xlMinimum** and **xlMaximum** can have different meanings, depending on the axis.

Crosses Property Example

This example sets the value axis in Chart1 to cross the category axis at the maximum x value.

```
Charts("Chart1").Axes(xlCategory).Crosses = xlMaximum
```

CrossesAt Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCrossesAtC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCrossesAtX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCrossesAtA "}

Returns or sets the point on the value axis where the category axis crosses it. Applies only to the value axis. Read/write **Double**.

Remarks

Setting this property causes the **Crosses** property to change to **xlAxisCrossesCustom**.

This property cannot be used on 3-D charts or radar charts.

CrossesAt Property Example

This example sets the category axis in Chart1 to cross the value axis at value 3.

```
With Charts("Chart1").Axes(xlValue)
    .Crosses = xlCustom
    .CrossesAt = 3
End With
```

DepthPercent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDepthPercentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDepthPercentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDepthPercentA "}

Returns or sets the depth of a 3-D chart as a percentage of the chart width (between 20 and 2000 percent). Read/write **Long**.

DepthPercent Property Example

This example sets the depth of Chart1 to be 50 percent of its width. The example should be run on a 3-D chart (the **DepthPercent** property fails on 2-D charts).

```
Charts("Chart1").DepthPercent = 50
```

DoughnutGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDoughnutGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDoughnutGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDoughnutGroupsA "}

On a 2-D chart, returns an object that represents either a single doughnut chart group (a **ChartGroup** object, Syntax 1) or a collection of the doughnut chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**DoughnutGroups(*Index*)**

Syntax 2

expression.**DoughnutGroups**

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

Index Optional **Variant**. Specifies the chart group.

DoughnutGroups Method Example

This example sets the starting angle for doughnut group one in Chart1.

```
Charts("Chart1").DoughnutGroups(1).FirstSliceAngle = 45
```

Elevation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproElevationC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproElevationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproElevationA "}

Returns or sets the elevation of the 3-D chart view, in degrees. Read/write **Long**.

Remarks

The chart elevation is the height at which you view the chart, in degrees. The default is 15 for most chart types. The value of this property must be between -90 and 90, except for 3-D bar charts, where it must be between 0 and 44.

Elevation Property Example

This example sets the chart elevation of Chart1 to 34 degrees. The example should be run on a 3-D chart (the **Elevation** property fails on 2-D charts).

```
Charts("Chart1").Elevation = 34
```

Floor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFloorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFloorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFloorA "}

Returns a **Floor** object that represents the floor of the 3-D chart. Read-only.

For information about using the **Floor** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Floor Property Example

This example sets the floor color of Chart1 to blue. The example should be run on a 3-D chart (the **Floor** property fails on 2-D charts).

```
Charts("Chart1").Floor.Interior.ColorIndex = 5
```

Font Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFontC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFontA "}

Returns a **Font** object that represents the font of the specified object. Read-only.

Font Property Example

This example sets the font in cell B5 on Sheet1 to 14-point bold italic.

```
With Worksheets("Sheet1").Range("B5").Font
    .Size = 14
    .Bold = True
    .Italic = True
End With
```

GapDepth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGapDepthC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGapDepthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGapDepthA "}

Returns or sets the distance between the data series in a 3-D chart, as a percentage of the marker width. The value of this property must be between 0 and 500. Read/write **Long**.

GapDepth Property Example

This example sets the distance between the data series in Chart1 to 200 percent of the marker width. The example should be run on a 3-D chart (the **GapDepth** property fails on 2-D charts).

```
Charts("Chart1").GapDepth = 200
```

GapWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGapWidthC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGapWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGapWidthA "}

Bar and Column charts: Returns or sets the space between bar or column clusters, as a percentage of the bar or column width. The value of this property must be between 0 and 500. Read/write **Long**.

Pie of Pie and Bar of Pie charts: Returns or sets the space between the primary and secondary sections of the chart. The value of this property must be between 5 and 200. Read/write **Long**.

GapWidth Property Example

This example sets the space between column clusters in Chart1 to be 50 percent of the column width.

```
Charts("Chart1").ChartGroups(1).GapWidth = 50
```

HasMajorGridlines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasMajorGridlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasMajorGridlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasMajorGridlinesA "}

True if the axis has major gridlines. Only axes in the primary axis group can have gridlines. Read/write **Boolean**.

HasMajorGridlines Property Example

This example sets the color of the major gridlines for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
   If .HasMajorGridlines Then
      .MajorGridlines.Border.ColorIndex = 3    'set color to red
   End If
End With
```

HasMinorGridlines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasMinorGridlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasMinorGridlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasMinorGridlinesA "}

True if the axis has minor gridlines. Only axes in the primary axis group can have gridlines. Read/write **Boolean**.

HasMinorGridlines Property Example

This example sets the color of the minor gridlines for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
   If .HasMinorGridlines Then
      .MinorGridlines.Border.ColorIndex = 4    'set color to green
   End If
End With
```

Insert Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthInsertC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthInsertX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthInsertA "}

Syntax 1 (**Range** object): Inserts a cell or a range of cells into the worksheet or macro sheet and shifts other cells away to make space.

Syntax 2 (**Characters** object): Inserts a string preceding the selected characters.

Syntax 1

expression.**Insert(*Shift*)**

Syntax 2

expression.**Insert(*String*)**

expression Required. An expression that returns a **Characters** or **Range** object.

Shift Optional **Variant**. Specifies which way to shift the cells. Can be one of the following **XlInsertShiftDirection** constants: **xlShiftToRight** or **xlShiftDown**. If this argument is omitted, Microsoft Excel decides based on the shape of the range.

String Required **String**. The string to insert.

Insert Method Example

This example inserts a new row before row four on Sheet1.

```
Worksheets("Sheet1").Rows(4).Insert
```

This example inserts new cells at the range A1:C5 on Sheet1 and shifts cells downward.

```
Worksheets("Sheet1").Range("A1:C5").Insert shift:=xlShiftDown
```

This example inserts a new row at the active cell. The example must be run from a worksheet.

```
ActiveCell.EntireRow.Insert
```

MajorGridlines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMajorGridlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMajorGridlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMajorGridlinesA "}

Returns a **Gridlines** object that represents the major gridlines for the specified axis. Only axes in the primary axis group can have gridlines. Read-only.

MajorGridlines Property Example

This example sets the color of the major gridlines for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
   If .HasMajorGridlines Then
      .MajorGridlines.Border.ColorIndex = 5    'set color to blue
   End If
End With
```

MajorTickMark Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMajorTickMarkC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMajorTickMarkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMajorTickMarkA "}

Returns or sets the type of major tick mark for the specified axis. Can be one of the following **XlTickMark** constants: **xlTickMarkNone**, **xlTickMarkInside**, **xlTickMarkOutside**, or **xlTickMarkCross**. Read/write **Long**.

MajorTickMark Property Example

This example sets the major tick marks for the value axis in Chart1 to be outside the axis.

```
Charts("Chart1").Axes(xlValue).MajorTickMark = xlTickMarkOutside
```

MajorUnit Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMajorUnitC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMajorUnitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMajorUnitA "}

Returns or sets the major units for the axis. Read/write **Double**.

Remarks

Setting this property sets the **MajorUnitIsAuto** property to **False**.

Use the **TickMarkSpacing** property to set tick mark spacing on the category axis.

MajorUnit Property Example

This example sets the major and minor units for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MajorUnit = 100
    .MinorUnit = 20
End With
```

MajorUnitIsAuto Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMajorUnitIsAutoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMajorUnitIsAutoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMajorUnitIsAutoA "}

True if Microsoft Excel calculates the major units for the axis. Read/write **Boolean**.

Remarks

Setting the **MajorUnit** property sets this property to **False**.

MajorUnitIsAuto Property Example

This example automatically sets the major and minor units for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MajorUnitIsAuto = True
    .MinorUnitIsAuto = True
End With
```

MaximumScale Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMaximumScaleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMaximumScaleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMaximumScaleA "}

Returns or sets the maximum value on the axis. Read/write **Double**.

Remarks

Setting this property sets the **MaximumScaleIsAuto** property to **False**.

MaximumScale Property Example

This example sets the minimum and maximum values for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MinimumScale = 10
    .MaximumScale = 120
End With
```

MaximumScaleIsAuto Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMaximumScaleIsAutoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMaximumScaleIsAutoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMaximumScaleIsAutoA "}

True if Microsoft Excel calculates the maximum value for the axis. Read/write **Boolean**.

Remarks

Setting the **MaximumScale** property sets this property to **False**.

MaximumScaleIsAuto Property Example

This example automatically calculates the minimum scale and the maximum scale for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MinimumScaleIsAuto = True
    .MaximumScaleIsAuto = True
End With
```

MinimumScale Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinimumScaleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinimumScaleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinimumScaleA "}

Returns or sets the minimum value on the axis. Read/write **Double**.

Remarks

Setting this property sets the **MinimumScaleIsAuto** property to **False**.

MinimumScale Property Example

This example sets the minimum and maximum values for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MinimumScale = 10
    .MaximumScale = 120
End With
```

MinimumScaleIsAuto Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinimumScaleIsAutoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinimumScaleIsAutoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinimumScaleIsAutoA "}

True if Microsoft Excel calculates the minimum value for the axis. Read/write **Boolean**.

Remarks

Setting the **MinimumScale** property sets this property to **False**.

MinimumScaleIsAuto Property Example

This example automatically calculates the minimum scale and the maximum scale for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MinimumScaleIsAuto = True
    .MaximumScaleIsAuto = True
End With
```

MinorGridlines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinorGridlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinorGridlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinorGridlinesA "}

Returns a **Gridlines** object that represents the minor gridlines for the specified axis. Only axes in the primary axis group can have gridlines. Read-only.

MinorGridlines Property Example

This example sets the color of the minor gridlines for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
   If .HasMinorGridlines Then
      .MinorGridlines.Border.ColorIndex = 5    'set color to blue
   End If
End With
```

MinorTickMark Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinorTickMarkC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinorTickMarkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinorTickMarkA "}

Returns or sets the type of minor tick mark for the specified axis. Can be one of the following **XlTickMark** constants: **xlTickMarkNone**, **xlTickMarkInside**, **xlTickMarkOutside**, or **xlTickMarkCross**. Read/write **Long**.

MinorTickMark Property Example

This example sets the minor tick marks for the value axis in Chart1 to be inside the axis.

```
Charts("Chart1").Axes(xlValue).MinorTickMark = xlTickMarkInside
```

MinorUnit Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinorUnitC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinorUnitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinorUnitA "}

Returns or sets the minor units on the axis. Read/write **Double**.

Remarks

Setting this property sets the **MinorUnitIsAuto** property to **False**.

Use the **TickMarkSpacing** property to set tick mark spacing on the category axis.

MinorUnit Property Example

This example sets the major and minor units for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MajorUnit = 100
    .MinorUnit = 20
End With
```

MinorUnitIsAuto Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinorUnitIsAutoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinorUnitIsAutoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinorUnitIsAutoA "}

True if Microsoft Excel calculates minor units for the axis. Read/write **Boolean**.

Remarks

Setting the **MinorUnit** property sets this property to **False**.

MinorUnitIsAuto Property Example

This example automatically calculates major and minor units for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
    .MajorUnitIsAuto = True
    .MinorUnitIsAuto = True
End With
```

ReversePlotOrder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReversePlotOrderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReversePlotOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReversePlotOrderA "}

True if Microsoft Excel plots data points from last to first. Read/write **Boolean**.

Remarks

This property cannot be used on radar charts.

ReversePlotOrder Property Example

This example plots data points from last to first on the value axis on Chart1.

```
Charts("Chart1").Axes(xlValue).ReversePlotOrder = True
```

ScaleType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproScaleTypeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproScaleTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproScaleTypeA "}

Returns or sets the value axis scale type. Can be one of the following **XlScaleType** constants: **xlScaleLinear** or **xlScaleLogarithmic**. Applies only to the value axis. Read/write **Long**.

Remarks

A logarithmic scale uses base 10 logarithms.

ScaleType Property Example

This example sets the value axis in Chart1 to use a logarithmic scale.

```
Charts("Chart1").Axes(xlValue).ScaleType = xlScaleLogarithmic
```

TickLabelPosition Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTickLabelPositionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTickLabelPositionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTickLabelPositionA "}

Describes the position of tick-mark labels on the specified axis. Can be one of the following **XlTickLabelPosition** constants: **xlTickLabelPositionNone**, **xlTickLabelPositionLow**, **xlTickLabelPositionHigh**, or **xlTickLabelPositionNextToAxis**. Read/write **Long**.

TickLabelPosition Property Example

This example sets tick-mark labels on the category axis in Chart1 to the high position (above the chart).

```
Charts("Chart1").Axes(xlCategory) _
    .TickLabelPosition = xlTickLabelPositionHigh
```

TickLabels Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTickLabelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTickLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTickLabelsA "}

Returns a **TickLabels** object that represents the tick-mark labels for the specified axis. Read-only.

TickLabels Property Example

This example sets the color of the tick-mark label font for the value axis in Chart1.

```
Charts("Chart1").Axes(xlValue).TickLabels.Font.ColorIndex = 3
```

TickLabelSpacing Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTickLabelSpacingC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTickLabelSpacingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTickLabelSpacingA "}

Returns or sets the number of categories or series between tick-mark labels. Applies only to category and series axes. Read/write **Long**.

Remarks

Tick-mark label spacing on the value axis is always calculated by Microsoft Excel.

TickLabelSpacing Property Example

This example sets the number of categories between tick-mark labels on the category axis in Chart1.

```
Charts("Chart1").Axes(xlCategory).TickLabelSpacing = 10
```

TickMarkSpacing Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTickMarkSpacingC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTickMarkSpacingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTickMarkSpacingA "}

Returns or sets the number of categories or series between tick marks. Applies only to category and series axes. Read/write **Long**.

Remarks

Use the **MajorUnit** and **MinorUnit** properties to set tick-mark spacing on the value axis.

TickMarkSpacing Property Example

This example sets the number of categories between tick marks on the category axis in Chart1.

```
Charts("Chart1").Axes(xlCategory).TickMarkSpacing = 10
```

AutoText Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoTextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoTextA "}

True if the object automatically generates appropriate text based on context. Read/write **Boolean**.

AutoText Property Example

This example sets the data labels for series one in Chart1 to automatically generate appropriate text.

```
Charts("Chart1").SeriesCollection(1).DataLabels.AutoText = True
```

Border Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBorderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBorderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBorderA "}

Returns a **Border** object that represents the border of the object. Read-only.

Border Property Example

This example sets the color of the chart area border of Chart1 to red.

```
Charts("Chart1").ChartArea.Border.ColorIndex = 3
```

BottomRightCell Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBottomRightCellC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBottomRightCellX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBottomRightCellA "}

Returns a **Range** object that represents the cell that lies under the the lower-right corner of the object. Read-only.

BottomRightCell Property Example

This example displays the address of the cell beneath the lower-right corner of embedded chart one on Sheet1.

```
MsgBox "The bottom right corner is over cell " & _
   Worksheets("Sheet1").ChartObjects(1).BottomRightCell.Address
```

BringToFront Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthBringToFrontC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthBringToFrontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthBringToFrontA "}

Brings the object to the front of the z-order.

Syntax

expression.**BringToFront**

expression Required. An expression that returns an object in the Applies To list.

BringToFront Method Example

This example brings embedded chart one on Sheet1 to the front of the z-order.

```
Worksheets("Sheet1").ChartObjects(1).BringToFront
```

Chart Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChartC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChartX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChartA "}

Returns a **Chart** object that represents the chart contained in the object. Read-only.

Chart Property Example

This example adds a title to the first embedded chart on Sheet1.

```
With Worksheets("Sheet1").ChartObjects(1).Chart
    .HasTitle = True
    .ChartTitle.Text = "1995 Rainfall Totals by Month"
End With
```

ClearFormats Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearFormatsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearFormatsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearFormatsA "}

Clears the formatting of the object.

Syntax

expression.**ClearFormats**

expression Required. An expression that returns an object in the Applies To list.

ClearFormats Method Example

This example clears all formatting from cells A1:G37 on Sheet1.

```
Worksheets("Sheet1").Range("A1:G37").ClearFormats
```

This example clears the formatting from embedded chart one on Sheet1.

```
Worksheets("Sheet1").ChartObjects(1).Chart.ChartArea.ClearFormats
```

Close Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCloseC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCloseX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCloseA "}

Closes the object. The **Workbooks** collection uses Syntax 1. **Window** and **Workbook** objects use Syntax 2.

Syntax 1

expression.**Close**

Syntax 2

expression.**Close(*SaveChanges, FileName, RouteWorkbook*)**

expression Required. An expression that returns an object in the Applies To list.

SaveChanges Optional **Variant**. If there are no changes to the workbook, this argument is ignored. If there are changes to the workbook and the workbook appears in other open windows, this argument is ignored. If there are changes to the workbook but the workbook doesn't appear in any other open windows, this argument specifies whether changes should be saved, as shown in the following table.

Value	Action
True	Saves the changes to the workbook. If there is not yet a file name associated with the workbook, then ***FileName*** is used. If ***FileName*** is omitted, the user is asked to supply a file name.
False	Does not save the changes to this file.
Omitted	Displays a dialog box asking the user whether or not to save changes.

FileName Optional **Variant**. Save changes under this file name.

RouteWorkbook Optional **Variant**. If the workbook doesn't need to be routed to the next recipient (if it has no routing slip or has already been routed), this argument is ignored. Otherwise, Microsoft Excel routes the workbook as shown in the following table.

Value	Meaning
True	Sends the workbook to the next recipient.
False	Doesn't send the workbook.
Omitted	Displays a dialog box asking the user whether the workbook should be sent.

Remarks

Closing a workbook from Visual Basic doesn't run any Auto_Close macros in the workbook. Use the **RunAutoMacros** method to run the auto close macros.

Close Method Example

This example closes Book1.xls and discards any changes that have been made to it.

```
Workbooks("BOOK1.XLS").Close SaveChanges:=False
```

This example closes all open workbooks. If there are changes in any open workbook, Microsoft Excel displays the appropriate prompts and dialog boxes for saving changes.

```
Workbooks.Close
```

Color Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColorA "}

Returns or sets the primary color of the object, as shown in the following table. Use the **RGB** function to create a color value. Read/write **Long**.

Object	Color
Border	The color of the border.
Borders	The color of all four borders of a range. If they're not all the same color, **Color** returns 0 (zero).
Font	The color of the font.
Interior	The cell shading color or the drawing object fill color.

Color Property Example

This example sets the color of the tick-mark labels on the value axis in Chart1.

```
Charts("Chart1").Axes(xlValue).TickLabels.Font.Color = RGB(0, 255, 0)
```

Count Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCountC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCountA "}

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

For information about using the **Count** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Count Property Example

This example displays the number of columns in the selection on Sheet1. The code also tests for a multiple-area selection; if one exists, the code loops on the areas of the multiple-area selection.

```
Worksheets("Sheet1").Activate
areaCount = Selection.Areas.Count
If areaCount <= 1 Then
   MsgBox "The selection contains " & _
      Selection.Columns.Count & " columns."
Else
   For i = 1 To areaCount
      MsgBox "Area " & i & " of the selection contains " & _
         Selection.Areas(i).Columns.Count & " columns."
   Next i
End If
```

This example makes the last character in cell A1 a superscript character.

```
n = Worksheets("Sheet1").Range("A1").Characters.Count
Worksheets("Sheet1").Range("A1").Characters(n, 1) _
   .Font.Superscript = True
```

CustomListCount Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCustomListCountC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCustomListCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCustomListCountA "}

Returns the number of defined custom lists (including built-in lists). Read-only **Long**.

CustomListCount Property Example

This example displays the number of custom lists that are currently defined.

```
MsgBox "There are currently " & Application.CustomListCount & _
   " defined custom lists."
```

DefaultFilePath Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDefaultFilePathC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDefaultFilePathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDefaultFilePathA "}

Returns or sets the default path that Microsoft Excel uses when it opens files. Read/write **String**.

DefaultFilePath Property Example

This example displays the current default file path.

```
MsgBox "The current default file path is " & _
   Application.DefaultFilePath
```

Delete Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDeleteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDeleteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDeleteA "}

Deletes the object. Syntax 2 applies only to **Range** objects.

Syntax 1

expression.**Delete**

Syntax 2

expression.**Delete(*Shift*)**

expression Required. An expression that returns an object in the Applies To list.

Shift Optional **Variant**. Used only with **Range** objects. Specifies how to shift cells to replace deleted cells. Can be one of the following **XlDeleteShiftDirection** constants: **xlShiftToLeft** or **xlShiftUp**. If this argument is omitted, Microsoft Excel decides based on the shape of the range.

Remarks

Deleting a **Point** or **LegendKey** object deletes the entire series.

You can delete custom document properties, but you cannot delete a built-in document property.

Delete Method Example

This example deletes cells A1:D10 on Sheet1 and shifts the remaining cells to the left.

```
Worksheets("Sheet1").Range("A1:D10").Delete Shift:=xlShiftToLeft
```

This example deletes every worksheet in the active workbook without displaying the confirmation dialog box.

```
Application.DisplayAlerts = False
For Each w In Worksheets
   w.Delete
Next w
Application.DisplayAlerts = True
```

This example sorts the data in the first column on Sheet1 and then deletes rows that contain duplicate data.

```
Worksheets("Sheet1").Range("A1").Sort _
        key1:=Worksheets("Sheet1").Range("A1")
Set currentCell = Worksheets("Sheet1").Range("A1")
Do While Not IsEmpty(currentCell)
    Set nextCell = currentCell.Offset(1, 0)
    If nextCell.Value = currentCell.Value Then
        currentCell.EntireRow.Delete
    End If
    Set currentCell = nextCell
Loop
```

DisplayBlanksAs Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayBlanksAsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayBlanksAsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayBlanksAsA "}

Returns or sets the way that blank cells are plotted on a chart. Can be one of the following **XlDisplayBlanksAs** constants: **xlNotPlotted**, **xlInterpolated**, or **xlZero**. Read/write **Long**.

DisplayBlanksAs Property Example

This example sets Microsoft Excel to not plot blank cells in Chart1.

```
Charts("Chart1").DisplayBlanksAs = xlNotPlotted
```

DisplayEquation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayEquationC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayEquationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayEquationA "}

True if the equation for the trendline is displayed on the chart (in the same data label as the R-squared value). Setting this property to **True** automatically turns on data labels. Read/write **Boolean**.

DisplayEquation Property Example

This example displays the R-squared value and equation for trendline one in Chart1. The example should be run on a 2-D column chart that has a trendline for the first series.

```
With Charts("Chart1").SeriesCollection(1).Trendlines(1)
    .DisplayRSquared = True
    .DisplayEquation = True
End With
```

DisplayExcel4Menus Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayExcel4MenusC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayExcel4MenusX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayExcel4MenusA "}

True if Microsoft Excel displays version 4.0 menu bars. Read/write **Boolean**.

DisplayExcel4Menus Property Example

This example switches the display to Microsoft Excel version 4.0 menus.

```
Application.DisplayExcel4Menus = True
```

DisplayFullScreen Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayFullScreenC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayFullScreenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayFullScreenA "}

True if Microsoft Excel is in full-screen mode. Read/write **Boolean**.

Remarks

Full-screen mode maximizes the application window so that it fills the entire screen and hides the application title bar (in Microsoft Windows). Toolbars, the status bar, and the formula bar maintain separate display settings for full-screen mode and normal mode.

DisplayFullScreen Property Example

This example sets Microsoft Excel to be displayed in full-screen mode.

```
Application.DisplayFullScreen = True
```

DisplayRecentFiles Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayRecentFilesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayRecentFilesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayRecentFilesA "}

True if the list of recently used files is displayed on the **File** menu. Read/write **Boolean**.

DisplayRecentFiles Property Example

This example turns off the list of recently used files.

```
Application.DisplayRecentFiles = False
```

DisplayRSquared Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayRSquaredC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayRSquaredX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayRSquaredA "}

True if the R-squared value of the trendline is displayed on the chart (in the same data label as the equation). Setting this property to **True** automatically turns on data labels. Read/write **Boolean**.

DisplayRSquared Property Example

This example displays the R-squared value and equation for trendline one in Chart1. The example should be run on a 2-D column chart that has a trendline for the first series.

```
With Charts("Chart1").SeriesCollection(1).Trendlines(1)
    .DisplayRSquared = True
    .DisplayEquation = True
End With
```

Duplicate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDuplicateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDuplicateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDuplicateA "}

Duplicates the object and returns a reference to the new copy.

Syntax

expression.**Duplicate**

expression Required. An expression that returns an object in the Applies To list.

Duplicate Method Example

This example duplicates embedded chart one on Sheet1 and then selects the copy.

```
Set dChart = Worksheets("Sheet1").ChartObjects(1).Duplicate
dChart.Select
```

FindFile Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFindFileC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFindFileX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFindFileA "}

Displays the **Open** dialog box.

Syntax

expression.**FindFile**

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

Remarks

This method displays the **Open** dialog box and allows the user to open a file. If a new file is opened successfully, this method returns **True**. If the user cancels the dialog box, this method returns **False**.

FindFile Method Example

This example displays the **Open** dialog box.

```
Application.FindFile
```

Forward Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproForwardC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproForwardX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproForwardA "}

Returns or sets the number of periods (or units on a scatter chart) that the trendline extends forward. Read/write **Long**.

Forward Property Example

This example sets the number of units that the trendline on Chart1 extends forward and backward. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
With Charts("Chart1").SeriesCollection(1).Trendlines(1)
    .Forward = 5
    .Backward = .5
End With
```

Function Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFunctionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFunctionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFunctionA "}

Returns or sets the function used to summarize the pivot field (data fields only). Can be one of the following **XlConsolidationFunction** constants: **xlAverage**, **xlCount**, **xlCountNums**, **xlMax**, **xlMin**, **xlProduct**, **xlStDev**, **xlStDevP**, **xlSum**, **xlVar**, or **xlVarP**. Read/write **Long**.

Function Property Example

This example sets the Sum of 1994 field in PivotTable1 to use the SUM function.

```
ActiveSheet.PivotTables("PivotTable1") _
    .PivotFields("Sum of 1994").Function = xlSum
```

HasLegend Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasLegendC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasLegendX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasLegendA "}

True if the chart has a legend. Read/write **Boolean**.

HasLegend Property Example

This example turns on the legend for Chart1 and then sets the legend font color to blue.

```
With Charts("Chart1")
    .HasLegend = True
    .Legend.Font.ColorIndex = 5
End With
```

HasTitle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasTitleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasTitleA "}

True if the axis or chart has a visible title. Read/write **Boolean**.

Remarks

An axis title is represented by an **AxisTitle** object.

A chart title is represented by a **ChartTitle** object.

HasTitle Property Example

This example adds an axis label to the category axis in Chart1.

```
With Charts("Chart1").Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Text = "July Sales"
End With
```

HeightPercent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHeightPercentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHeightPercentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHeightPercentA "}

Returns or sets the height of a 3-D chart as a percentage of the chart width (between 5 and 500 percent). Read/write **Long**.

HeightPercent Property Example

This example sets the height of Chart1 to 80 percent of its width. The example should be run on a 3-D chart.

```
Charts("Chart1").HeightPercent = 80
```

Intercept Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInterceptC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInterceptX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInterceptA "}

Returns or sets the point where the trendline crosses the value axis. Read/write **Double**.

For information about using the **Intercept** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Remarks

Setting this property sets the **InterceptIsAuto** property to **False**.

Intercept Property Example

This example sets trendline one in Chart1 to cross the value axis at 5. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
Charts("Chart1").SeriesCollection(1).Trendlines(1).Intercept = 5
```

InterceptIsAuto Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInterceptIsAutoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInterceptIsAutoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInterceptIsAutoA "}

True if the point where the trendline crosses the value axis is automatically determined by the regression. Read/write **Boolean**.

Remarks

Setting the **Intercept** property sets this property to **False**.

InterceptIsAuto Property Example

This example sets Microsoft Excel to automatically determine the trendline intercept point for Chart1. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
Charts("Chart1").SeriesCollection(1).Trendlines(1) _
    .InterceptIsAuto = True
```

Legend Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLegendC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLegendX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLegendA "}

Returns a **Legend** object that represents the legend for the chart. Read-only.

Legend Property Example

This example turns on the legend for Chart1 and then sets the legend font color to blue.

```
Charts("Chart1").HasLegend = True
Charts("Chart1").Legend.Font.ColorIndex = 5
```

Line3DGroup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLine3DGroupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLine3DGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLine3DGroupA "}

Returns a **ChartGroup** object that represents the line chart group on a 3-D chart. Read-only.

Line3DGroup Property Example

This example sets the 3-D line group in Chart1 to use a different color for each data marker.

```
Charts("Chart1").Line3DGroup.VaryByCategories = True
```

LineGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthLineGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthLineGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthLineGroupsA "}

On a 2-D chart, returns an object that represents either a single line chart group (a **ChartGroup** object, Syntax 1) or a collection of the line chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**LineGroups(*Index*)**

Syntax 2

expression.**LineGroups**

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

Index Optional **Variant**. Specifies the chart group.

LineGroups Method Example

This example sets line group one in Chart1 to use a different color for each data marker. The example should be run on a 2-D chart.

```
Charts("Chart1").LineGroups(1).VaryByCategories = True
```

MacroType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMacroTypeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMacroTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMacroTypeA "}

Returns or sets what the name refers to. Read-write **Long**.

Can be one of the following **XlMacroType** constants.

Constant	Meaning
xlCommand	The name refers to a user-defined macro.
xlFunction	The name refers to a user-defined function.
xlNotXLM	The name doesn't refer to a function or macro.

MacroType Property Example

This example assumes that you created a custom function or command on a Microsoft Excel version 4.0 macro sheet. The example displays the function category, in the language of the macro. It assumes that the name of the custom function or command is the only name in the workbook.

```
With ActiveWorkbook.Names(1)
   If .MacroType <> xlNotXLM Then
      MsgBox "The category for this name is " & .Category
   Else
      MsgBox "This name does not refer to" & _
         " a custom function or command."
   End If
End With
```

NameIsAuto Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNameIsAutoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNameIsAutoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNameIsAutoA "}

True if Microsoft Excel automatically determines the name of the trendline. Read/write **Boolean**.

NameIsAuto Property Example

This example sets Microsoft Excel to automatically determine the name for trendline one in Chart1. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
Charts("Chart1").SeriesCollection(1).Trendlines(1).NameIsAuto = True
```

NumberFormatLinked Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNumberFormatLinkedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNumberFormatLinkedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNumberFormatLinkedA "}

True if the number format is linked to the cells (so that the number format changes in the labels when it changes in the cells). Read/write **Boolean**.

NumberFormatLinked Property Example

This example links the number format for tick-mark labels to its cells for the value axis in Chart1.

```
Charts("Chart1").Axes(xlValue).TickLabels.NumberFormatLinked = True
```

PageSetup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageSetupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageSetupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageSetupA "}

Returns a **PageSetup** object that contains all the page setup settings for the specified object. Read-only.

PageSetup Property Example

This example sets the center header text for Chart1.

```
Charts("Chart1").PageSetup.CenterHeader = "December Sales"
```

Perspective Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPerspectiveC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPerspectiveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPerspectiveA "}

Returns or sets the perspective for the 3-D chart view. Must be between 0 and 100. This property is ignored if the **RightAngleAxes** property is **True**. Read/write **Long**.

Perspective Property Example

This example sets the perspective of Chart1 to 70. The example should be run on a 3-D chart.

```
Charts("Chart1").RightAngleAxes = False
Charts("Chart1").Perspective = 70
```

Pie3DGroup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPie3DGroupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPie3DGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPie3DGroupA "}

Returns a **ChartGroup** object that represents the pie chart group on a 3-D chart. Read-only.

Pie3DGroup Property Example

This example sets the 3-D pie group in Chart1 to use a different color for each data marker.

```
Charts("Chart1").Pie3DGroup.VaryByCategories = True
```

PieGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPieGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPieGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPieGroupsA "}

On a 2-D chart, returns an object that represents either a single pie chart group (a **ChartGroup** object, Syntax 1) or a collection of the pie chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**PieGroups(*Index*)**

Syntax 2

expression.**PieGroups**

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

Index Optional **Variant**. Specifies the chart group.

PieGroups Method Example

This example sets pie group one in Chart1 to use a different color for each data marker. The example should be run on a 2-D chart.

```
Charts("Chart1").PieGroups(1).VaryByCategories = True
```

Placement Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPlacementC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPlacementX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPlacementA "}

Returns or sets the way the object is attached to the cells below it. Can be one of the following **XlPlacement** constants: **xlMoveAndSize**, **xlMove**, or **xlFreeFloating**. Can be used only on objects on a worksheet. Read/write **Long**.

Placement Property Example

This example sets embedded chart one on Sheet1 to be free-floating (it neither moves nor is sized with its underlying cells).

```
Worksheets("Sheet1").ChartObjects(1).Placement = xlFreeFloating
```

PlotArea Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPlotAreaC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPlotAreaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPlotAreaA "}

Returns a **PlotArea** object that represents the plot area of a chart. Read-only.

PlotArea Property Example

This example sets the color of the plot area interior of Chart1 to cyan.

```
Charts("Chart1").PlotArea.Interior.ColorIndex = 8
```

PrintObject Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintObjectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintObjectA "}

True if the object will be printed when the document is printed. Read/write **Boolean**.

PrintObject Property Example

This example sets embedded chart one on Sheet1 to be printed with the worksheet.

```
Worksheets("Sheet1").ChartObjects(1).PrintObject = True
```

RadarGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRadarGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRadarGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRadarGroupsA "}

On a 2-D chart, returns an object that represents either a single radar chart group (a **ChartGroup** object, Syntax 1) or a collection of the radar chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**RadarGroups(*Index*)**

Syntax 2

expression.**RadarGroups**

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

Index Optional **Variant**. Specifies the chart group.

RadarGroups Method Example

This example sets radar group one in Chart1 to use a different color for each data marker. The example should be run on a 2-D chart.

```
Charts("Chart1").RadarGroups(1).VaryByCategories = True
```

RightAngleAxes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRightAngleAxesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRightAngleAxesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRightAngleAxesA "}

True if the chart axes are at right angles, independent of chart rotation or elevation. Applies only to 3-D line, column, and bar charts. Read/write **Boolean**.

Remarks

If this property is **True**, the **Perspective** property is ignored.

RightAngleAxes Property Example

This example sets the axes in Chart1 to intersect at right angles. The example should be run on a 3-D chart.

```
Charts("Chart1").RightAngleAxes = True
```

RoundedCorners Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRoundedCornersC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRoundedCornersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRoundedCornersA "}

True if the embedded chart has rounded corners. Read/write **Boolean**.

RoundedCorners Property Example

This example adds rounded corners to embedded chart one on Sheet1.

```
Worksheets("Sheet1").ChartObjects(1).RoundedCorners = True
```

SendToBack Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSendToBackC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSendToBackX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSendToBackA "}

Sends the object to the back of the z-order.

Syntax

expression.**SendToBack**

expression Required. An expression that returns an object in the Applies To list.

SendToBack Method Example

This example sends embedded chart one on Sheet1 to the back of the z-order.

```
Worksheets("Sheet1").ChartObjects(1).SendToBack
```

SeriesCollection Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSeriesCollectionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSeriesCollectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSeriesCollectionA "}

Returns an object that represents either a single series (a **Series** object, Syntax 1) or a collection of all the series (a **SeriesCollection** object, Syntax 2) in the chart or chart group.

Syntax 1

expression.**SeriesCollection(*Index*)**

Syntax 2

expression.**SeriesCollection**

expression Required. An expression that returns a **Chart** or **ChartGroup** object.

Index Optional **Variant**. The name or number of the series.

SeriesCollection Method Example

This example turns on data labels for series one in Chart1.

```
Charts("Chart1").SeriesCollection(1).HasDataLabels = True
```

SetDefaultChart Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetDefaultChartC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetDefaultChartX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetDefaultChartA "}

Specifies the name of the chart template that Microsoft Excel will use when creating new charts.

Syntax

expression.**SetDefaultChart(*FormatName*)**

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

FormatName Optional **Variant**. Specifies the name of a custom autoformat. This name can be a string naming a custom autoformat, or it can be the special constant **xlBuiltIn** to specify the built-in chart template.

SetDefaultChart Method Example

This example sets the default chart template to the custom autoformat named "Monthly Sales."

```
Application.SetDefaultChart FormatName:="Monthly Sales"
```

ShowLegendKey Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowLegendKeyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowLegendKeyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowLegendKeyA "}

True if the data label legend key is visible. Read/write **Boolean**.

ShowLegendKey Property Example

This example sets the data labels for series one in Chart1 to show values and the legend key.

```
With Charts("Chart1").SeriesCollection(1).DataLabels
    .ShowLegendKey = True
    .Type = xlShowValue
End With
```

Size Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSizeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSizeA "}

Returns or sets the size of the font. Read/write **Variant**.

Size Property Example

This example sets the font size for cells A1:D10 on Sheet1 to 12 points.

```
With Worksheets("Sheet1").Range("A1:D10")
    .Value = "Test"
    .Font.Size = 12
End With
```

SurfaceGroup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSurfaceGroupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSurfaceGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSurfaceGroupA "}

Returns a **ChartGroup** object that represents the surface chart group of a 3-D chart. Read-only.

SurfaceGroup Property Example

This example sets the 3-D surface group in Chart1 to use a different color for each data marker. The example should be run on a 3-D chart.

```
Charts("Chart1").SurfaceGroup.VaryByCategories = True
```

TopLeftCell Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTopLeftCellC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTopLeftCellX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTopLeftCellA "}

Returns a **Range** object that represents the cell that lies under the upper-left corner of the specified object. Read-only.

TopLeftCell Property Example

This example displays the address of the cell beneath the upper-left corner of embedded chart one on Sheet1.

```
MsgBox "The top left corner is over cell " & _
   Worksheets("Sheet1").ChartObjects(1).TopLeftCell.Address
```

Visible Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVisibleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVisibleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVisibleA "}

True if the object is visible. For a chart or worksheet, this property can be set to **xlVeryHidden**. This hides the object so that the only way for you to make it visible again is by setting this property to **True** (the user cannot make the object visible). Read/write **Boolean** or **Long**.

Remarks

The **Visible** property for a pivot item is **True** if the item is currently visible in the table.

If you set the **Visible** property for a name to **False**, the name won't appear in the **Define Name** dialog box.

Visible Property Example

This example hides Sheet1.

```
Worksheets("Sheet1").Visible = False
```

This example makes Sheet1 visible.

```
Worksheets("Sheet1").Visible = True
```

This example makes every sheet in the active workbook visible.

```
For Each sh In Sheets
   sh.Visible = True
Next sh
```

This example creates a new worksheet and then sets its **Visible** property to **xlVeryHidden**. To refer to the sheet, use its object variable, `newSheet`, as shown in the last line of the example. To use the `newSheet` object variable in another procedure, you must declare it as a public variable (`Public newSheet As Object`) in the first line of the module preceding any **Sub** or **Function** procedure.

```
Set newSheet = Worksheets.Add
newSheet.Visible = xlVeryHidden
newSheet.Range("A1:D4").Formula = "=RAND()"
```

Walls Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWallsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWallsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWallsA "}

Returns a **Walls** object that represents the walls of the 3-D chart. Read-only.

Remarks

This property doesn't apply to 3-D pie charts.

Walls Property Example

This example sets the color of the wall border of Chart1 to red. The example should be run on a 3-D chart.

```
Charts("Chart1").Walls.Border.ColorIndex = 3
```

Weight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWeightC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWeightA "}

Returns or sets the weight of the border. Can be one of the following **XlBorderWeight** constants: **xlHairline**, **xlThin**, **xlMedium**, or **xlThick**. Read/write **Long**.

Weight Property Example

This example sets the border weight for oval one on Sheet1.

```
Worksheets("Sheet1").Ovals(1).Border.Weight = xlMedium
```

XYGroups Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthXYGroupsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthXYGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthXYGroupsA "}

On a 2-D chart, returns an object that represents either a single scatter chart group (a **ChartGroup** object, Syntax 1) or a collection of the scatter chart groups (a **ChartGroups** collection, Syntax 2).

Syntax 1

expression.**XYGroups(*Index*)**

Syntax 2

expression.**XYGroups**

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

Index Optional **Variant**. Specifies the chart group.

XYGroups Method Example

This example sets X-Y group (scatter group) one to use a different color for each data marker. The example should be run on a 2-D chart.

```
Charts("Chart1").XYGroups(1).VaryByCategories = True
```

ZOrder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproZOrderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproZOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproZOrderA "}

Returns the z-order position of the object. Read-only **Long**.

Remarks

In any collection of objects, the object at the back of the z-order is *collection*(1), and the object at the front of the z-order is *collection*(*collection*.**Count**). For example, if there are embedded charts on the active sheet, the chart at the back of the z-order is `ActiveSheet.ChartObjects(1)`, and the chart at the front of the z-order is `ActiveSheet.ChartObjects(ActiveSheet.ChartObjects.Count)`.

ZOrder Property Example

This example displays the z-order position of embedded chart one on Sheet1.

```
MsgBox "The chart's z-order position is " & _
   Worksheets("Sheet1").ChartObjects(1).ZOrder
```

CopyObjectsWithCells Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCopyObjectsWithCellsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCopyObjectsWithCellsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCopyObjectsWithCellsA "}

True if objects are cut, copied, extracted, and sorted with cells. Read/write **Boolean**.

CopyObjectsWithCells Property Example

This example sets Microsoft Excel to cut, copy, extract, and sort objects with cells.

```
Application.CopyObjectsWithCells = True
```

CopyFromRecordset Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCopyFromRecordsetC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCopyFromRecordsetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCopyFromRecordsetA "}

Copies the contents of a DAO **Recordset** object onto a worksheet, beginning at the upper-left corner of the specified range. If the **Recordset** object contains fields with OLE objects in them, this method fails.

Syntax

expression.**CopyFromRecordset(*Data, MaxRows, MaxColumns*)**

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

Data Required **Void**. The **Recordset** object to copy into the range.

MaxRows Optional **Variant**. The maximum number of records to copy onto the worksheet. If this argument is omitted, all the records in the **Recordset** object are copied.

MaxColumns Optional **Variant**. The maximum number of fields to copy onto the worksheet. If this argument is omitted, all the fields in the **Recordset** object are copied.

Remarks

Copying begins at the current row of the **Recordset** object. After copying is completed, the **EOF** property of the **Recordset** object is **True**.

CopyFromRecordset Method Example

This example copies the field names from a DAO **Recordset** object into the first row of a worksheet and formats the names as bold. The example then copies the recordset onto the worksheet, beginning at cell A2.

```
For iCols = 0 to rs.Fields.Count - 1
   ws.Cells(1, iCols + 1).Value = rs.Fields(iCols).Name
Next
ws.Range(ws.Cells(1, 1),  _
   ws.Cells(1, rs.Fields.Count)).Font.Bold = True
ws.Range("A2").CopyFromRecordset rs
```

Cursor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCursorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCursorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCursorA "}

Returns or sets the appearance of the mouse pointer in Microsoft Excel. Read/write **Long**.

Can be one of the following **XlMousePointer** constants.

Constant	Description
xlDefault	The default pointer
xlWait	The hourglass pointer 
xlNorthwestArrow	The northwest-arrow pointer 
xlIBeam	The I-beam pointer 

Remarks

The **Cursor** property isn't reset automatically when the macro stops running. You should reset the pointer to **xlDefault** before your macro stops running.

Cursor Property Example

This example changes the mouse pointer to an I-beam, pauses, and then changes it to the default pointer.

```
Sub ChangeCursor()
   Application.Cursor = xlIBeam
   For x = 1 To 1000
      For y = 1 to 1000
      Next y
   Next x
   Application.Cursor = xlDefault
End Sub
```

EnableAutoFilter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableAutoFilterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableAutoFilterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableAutoFilterA "}

True if AutoFilter arrows are enabled when user-interface-only protection is turned on. Read/write **Boolean**.

Remarks

This property applies to each worksheet and isn't saved with the worksheet or session.

EnableAutoFilter Property Example

This example enables the AutoFilter arrows on a protected worksheet.

```
ActiveSheet.EnableAutoFilter = True
ActiveSheet.Protect contents:=True, userInterfaceOnly:=True
```

EnableOutlining Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableOutliningC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableOutliningX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableOutliningA "}

True if outlining symbols are enabled when user-interface-only protection is turned on. Read/write **Boolean**.

Remarks

This property applies to each worksheet and isn't saved with the worksheet or session.

EnableOutlining Property Example

This example enables outlining symbols on a protected worksheet.

```
ActiveSheet.EnableOutlining = True
ActiveSheet.Protect contents:=True, userInterfaceOnly:=True
```

EnablePivotTable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnablePivotTableC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnablePivotTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnablePivotTableA "}

True if PivotTable controls and actions are enabled when user-interface-only protection is turned on. Read/write **Boolean**.

Remarks

This property applies to each worksheet and isn't saved with the worksheet or session.

There must be a sufficient number of unlocked cells below and to the right of the PivotTable for Microsoft Excel to recalculate and display the PivotTable.

EnablePivotTable Property Example

This example enables PivotTable controls on a protected worksheet.

```
ActiveSheet.EnablePivotTable = True
ActiveSheet.Protect contents:=True, userInterfaceOnly:=True
```

RefersToRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefersToRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefersToRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefersToRangeA "}

Returns the **Range** object referred to by a **Name** object. Read-only.

Remarks

If the **Name** object doesn't refer to a range (for example, if it refers to a constant or a formula), this property fails.

To change the range that a name refers to, use the **RefersTo** property.

RefersToRange Property Example

This example displays the number of rows and columns in the print area on the active worksheet.

```
p = Names("Print_Area").RefersToRange.Value
MsgBox "Print_Area: " & UBound(p, 1) & " rows, " & _
   UBound(p, 2) & " columns"
```

Background Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBackgroundC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBackgroundA "}

Returns or sets the text background type. Can be one of the following **XlBackground** constants: **xlBackgroundAutomatic**, **xlBackgroundOpaque**, or **xlBackgroundTransparent**. This property is used only for text on charts. Read/write **Long**.

Background Property Example

This example adds a chart title to embedded chart one on Sheet1 and then sets the font size and background type for the title.

```
With Worksheets("Sheet1").ChartObjects(1).Chart
   .HasTitle = True
   .ChartTitle.Text = "1995 Rainfall Totals by Month"
   With .ChartTitle.Font
      .Size = 10
      .Background = xlBackgroundTransparent
   End With
End With
```

Bold Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBoldC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBoldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBoldA "}

True if the font is bold. Read/write **Variant**.

Bold Property Example

This example sets the font to bold for the range A1:A5 on Sheet1.

```
Worksheets("Sheet1").Range("A1:A5").Font.Bold = True
```

CategoryLocal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCategoryLocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCategoryLocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCategoryLocalA "}

Returns or sets the category for the specified name, in the language of the user, if the name refers to a custom function or command. Read/write **String**.

CategoryLocal Property Example

This example displays, in the language of the user, the function category of either a custom function or a command created on a Microsoft Excel 4.0 macro sheet. The example assumes that the custom function name or command name is the only name in the workbook.

```
With ActiveWorkbook.Names(1)
   If .MacroType <> xlNone Then
      MsgBox "The category for this name is " & .CategoryLocal
   Else
      MsgBox "This name does not refer to" & _
         " a custom function or command."
   End If
End With
```

CentimetersToPoints Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCentimetersToPointsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCentimetersToPointsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCentimetersToPointsA "}

Converts a measurement from centimeters to points (one point equals 0.035 centimeters).

Syntax

expression.**CentimetersToPoints**(***Centimeters***)

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

Centimeters Required **Double**. Specifies the centimeter value to be converted to points.

CentimetersToPoints Method Example

This example sets the left margin of Sheet1 to 5 centimeters.

```
Worksheets("Sheet1").PageSetup.LeftMargin = _
      Application.CentimetersToPoints(5)
```

ChartObjects Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthChartObjectsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthChartObjectsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthChartObjectsA "}

Returns an object that represents either a single embedded chart (a **ChartObject** object, Syntax 1) or a collection of all the embedded charts (a **ChartObjects** object, Syntax 2) on the sheet.

Syntax 1

expression.**ChartObjects(*Index*)**

Syntax 2

expression.**ChartObjects**

expression Required. An expression that returns an object in the Applies To list. If you specify a **Chart** object, it must be a chart sheet (it cannot be an embedded chart).

Index Opional **Variant**. The name or number of the chart. This argument can be an array, to specify more than one chart.

Remarks

This method isn't equivalent to the **Charts** property. This method returns embedded charts; the **Charts** property returns chart sheets. Use the **Chart** property to return the **Chart** object for an embedded chart.

ChartObjects Method Example

This example adds a title to embedded chart one on Sheet1.

```
With Worksheets("Sheet1").ChartObjects(1).Chart
    .HasTitle = True
    .ChartTitle.Text = "1995 Rainfall Totals by Month"
End With
```

This example creates a new series in embedded chart one on Sheet1. The data source for the new series is the range B1:B10 on Sheet1.

```
Worksheets("Sheet1").ChartObjects(1).Activate
ActiveChart.SeriesCollection.Add _
    source:=Worksheets("Sheet1").Range("B1:B10")
```

This example clears the formatting of embedded chart one on Sheet1.

```
Worksheets("Sheet1").ChartObjects(1).Chart.ChartArea.ClearFormats
```

CircularReference Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCircularReferenceC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCircularReferenceX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCircularReferenceA "}

Returns a **Range** object that represents the range containing the first circular reference on the sheet, or returns **Nothing** if there's no circular reference on the sheet. The circular reference must be removed before calculation can proceed. Read-only.

CircularReference Property Example

This example selects the first cell in the first circular reference on Sheet1.

```
Worksheets("Sheet1").CircularReference.Select
```

ColorIndex Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColorIndexC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColorIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColorIndexA "}

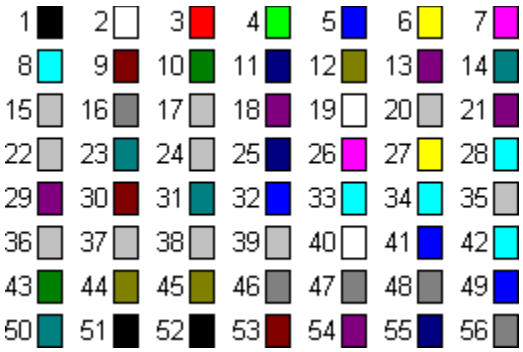
Returns or sets the color of the border, font, or interior, as shown in the following table. The color is specified as an index value into the current color palette, or as one of the following **XlColorIndex** constants: **xlColorIndexAutomatic** or **xlColorIndexNone**. Read/write **Variant**.

Object	ColorIndex
Border	The color of the border.
Borders	The color of all four borders. Returns **Null** if all four borders aren't the same color.
Font	The color of the font. Specify **xlColorIndexAutomatic** to use the automatic color.
Interior	The color of the interior fill. Set this property to **xlColorIndexNone** to specify that you don't want an interior fill. Set this property to **xlColorIndexAutomatic** to specify the automatic fill (for drawing objects).

Remarks

This property specifies a color as an index into the workbook color palette. You can use the **Colors** method to return the current color palette.

The following illustration shows the color-index values in the default color palette.



ColorIndex Property Example

The following examples assume that you're using the default color palette.

This example changes the font color in cell A1 on Sheet1 to red.

```
Worksheets("Sheet1").Range("A1").Font.ColorIndex = 3
```

This example sets the color of the major gridlines for the value axis in Chart1.

```
With Charts("Chart1").Axes(xlValue)
   If .HasMajorGridlines Then
      .MajorGridlines.Border.ColorIndex = 5    'set color to blue
   End If
End With
```

This example sets the color of the chart area interior of Chart1 to red and sets the border color to blue.

```
With Charts("Chart1").ChartArea
   .Interior.ColorIndex = 3
   .Border.ColorIndex = 5
End With
```

DataLabels Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDataLabelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDataLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDataLabelsA "}

Returns an object that represents either a single data label (a **DataLabel** object, Syntax 1) or a collection of all the data labels for the series (a **DataLabels** collection, Syntax 2).

Syntax 1

expression.**DataLabels(*Index*)**

Syntax 2

expression.**DataLabels**

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

Index Optional **Variant**. The number of the data label.

Remarks

If the series has the **Show Value** option turned on for the data labels, the returned collection can contain up to one label for each point. Data labels can be turned on or off for individual points in the series.

If the series is on an area chart and has the **Show Label** option turned on for the data labels, the returned collection contains only a single label, which is the label for the area series.

DataLabels Method Example

This example sets the data labels for series one in Chart1 to show their key, assuming that their values are visible when the example runs.

```
With Charts("Chart1").SeriesCollection(1)
    .HasDataLabels = True
    With .DataLabels
        .ShowLegendKey = True
        .Type = xlValue
    End With
End With
```

DisplayScrollBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayScrollBarsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayScrollBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayScrollBarsA "}

True if scroll bars are visible for all workbooks. Read/write **Boolean**.

DisplayScrollBars Property Example

This example turns off scroll bars for all workbooks.

```
Application.DisplayScrollBars = False
```

DownBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDownBarsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDownBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDownBarsA "}

Returns a **DownBars** object that represents the down bars on a line chart. Applies only to line charts. Read-only.

DownBars Property Example

This example turns on up bars and down bars for chart group one in Chart1 and then sets their colors. The example should be run on a 2-D line chart that has two series that cross each other at one or more data points.

```
With Charts("Chart1").ChartGroups(1)
    .HasUpDownBars = True
    .DownBars.Interior.ColorIndex = 3
    .UpBars.Interior.ColorIndex = 5
End With
```

DropLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDropLinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDropLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDropLinesA "}

Returns a **DropLines** object that represents the drop lines for a series on a line chart or area chart. Applies only to line charts or area charts. Read-only.

DropLines Property Example

This example turns on drop lines for chart group one in Chart1 and then sets their line style, weight, and color. The example should be run on a 2-D line chart that has one series.

```
With Charts("Chart1").ChartGroups(1)
   .HasDropLines = True
   With .DropLines.Border
      .LineStyle = xlThin
      .Weight = xlMedium
      .ColorIndex = 3
   End With
End With
```

Explosion Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproExplosionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproExplosionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproExplosionA "}

Returns or sets the explosion value for a pie-chart or doughnut-chart slice. Returns 0 (zero) if there's no explosion (the tip of the slice is in the center of the pie). Read/write **Long**.

Explosion Property Example

This example sets the explosion value for point two in Chart1. The example should be run on a pie chart.

```
Charts("Chart1").SeriesCollection(1).Points(2).Explosion = 20
```

FirstSliceAngle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFirstSliceAngleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFirstSliceAngleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFirstSliceAngleA "}

Returns or sets the angle of the first pie-chart or doughnut-chart slice, in degrees (clockwise from vertical). Applies only to pie, 3-D pie, and doughnut charts. Read/write **Long**.

FirstSliceAngle Property Example

This example sets the angle for the first slice in chart group one in Chart1. The example should be run on a 2-D pie chart.

```
Charts("Chart1").ChartGroups(1).FirstSliceAngle = 15
```

FormulaLocal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormulaLocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormulaLocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormulaLocalA "}

Returns or sets the formula for the object, using A1-style references in the language of the user. Read/write **Variant** for **Range** objects, read/write **String** for **Series** objects.

Remarks

If the cell contains a constant, this property returns that constant. If the cell is empty, the property returns an empty string. If the cell contains a formula, the property returns the formula as a string, in the same format in which it would be displayed in the formula bar (including the equal sign).

If you set the value or formula of a cell to a date, Microsoft Excel checks to see whether that cell is already formatted with one of the date or time number formats. If not, the number format is changed to the default short date number format.

If the range is a one- or two-dimensional range, you can set the formula to a Visual Basic array of the same dimensions. Similarly, you can put the formula into a Visual Basic array.

Setting the formula of a multiple-cell range fills all cells in the range with the formula.

FormulaLocal Property Example

Assume that you enter the formula =SUM(A1:A10) in cell A11 on worksheet one, using the American English version of Microsoft Excel. If you then open the workbook on a computer that's running the German version and run the following example, the example displays the formula =SUMME(A1:A10) in a message box.

```
MsgBox Worksheets(1).Range(A11).FormulaLocal
```

FormulaR1C1 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormulaR1C1C "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormulaR1C1X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormulaR1C1A "}

Returns or sets the formula for the object, using R1C1-style notation in the language of the macro. Read/write **Variant** for **Range** objects, read/write **String** for **Series** objects.

Remarks

If the cell contains a constant, this property returns the constant. If the cell is empty, the property returns an empty string. If the cell contains a formula, the property returns the formula as a string, in the same format in which it would be displayed in the formula bar (including the equal sign).

If you set the value or formula of a cell to a date, Microsoft Excel checks to see whether that cell is already formatted with one of the date or time number formats. If not, the number format is changed to the default short date number format.

If the range is a one- or two-dimensional range, you can set the formula to a Visual Basic array of the same dimensions. Similarly, you can put the formula into a Visual Basic array.

Setting the formula of a multiple-cell range fills all cells in the range with the formula.

FormulaR1C1 Property Example

This example sets the formula for cell B1 on Sheet1.

```
Worksheets("Sheet1").Range("B1").FormulaR1C1 = "=SQRT(R1C1)"
```

FormulaR1C1Local Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormulaR1C1LocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormulaR1C1LocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormulaR1C1LocalA "}

Returns or sets the formula for the object, using R1C1-style notation in the language of the user. Read/write **Variant** for **Range** objects, read/write **String** for **Series** objects.

Remarks

If the cell contains a constant, this property returns that constant. If the cell is empty, the property returns an empty string. If the cell contains a formula, the property returns the formula as a string, in the same format in which it would be displayed in the formula bar (including the equal sign).

If you set the value or formula of a cell to a date, Microsoft Excel checks to see whether that cell is already formatted with one of the date or time number formats. If not, the number format is changed to the default short date number format.

If the range is a one- or two-dimensional range, you can set the formula to a Visual Basic array of the same dimensions. Similarly, you can put the formula into a Visual Basic array.

Setting the formula of a multiple-cell range fills all cells in the range with the formula.

FormulaR1C1Local Property Example

Assume that you enter the formula =SUM(A1:A10) in cell A11 on worksheet one, using the American English version of Microsoft Excel. If you then open the workbook on a computer that's running the German version and run the following example, the example displays the formula =SUMME(Z1S1:Z10S1) in a message box.

```
MsgBox Worksheets(1).Range("A11").FormulaR1C1Local
```

GetOpenFilename Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGetOpenFilenameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGetOpenFilenameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGetOpenFilenameA "}

Displays the standard **Open** dialog box and gets a file name from the user without actually opening any files.

Syntax

expression.**GetOpenFilename(*FileFilter, FilterIndex, Title, ButtonText, MultiSelect*)**

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

FileFilter Optional **Variant**. A string specifying file filtering criteria.

In Windows, this string consists of pairs of file filter strings followed by the MS-DOS wildcard file filter specification, with each part and each pair separated by commas. Each separate pair is listed in the **Files of type** drop-down list box. For example, the following string specifies two file filters, text and addin: "Text Files (*.txt),*.txt,Add-In Files (*.xla),*.xla".

To use multiple MS-DOS wildcard expressions for a single file filter type, separate the wildcard expressions with semicolons; for example, "Visual Basic Files (*.bas; *.txt),*.bas;*.txt".

If omitted in Windows, this argument defaults to "All Files (*.*),*.*".

On the Macintosh, this string is a list of comma-separated file type codes (for example, "TEXT,XLA5,XLS4"). Spaces are significant and shouldn't be inserted before or after the comma separators unless they're part of the file type code. If omitted, this argument defaults to all file types.

FilterIndex Optional **Variant**. Windows only (ignored on the Macintosh). Specifies the index numbers of the default file filtering criteria, from 1 to the number of filters specified in ***FileFilter***. If this argument is omitted or greater than the number of filters present, the first file filter is used.

Title Optional **Variant**. Windows only (ignored on the Macintosh). Specifies the title of the dialog box. If this argument is omitted, the title is "Open."

ButtonText Optional **Variant**. Macintosh only (ignored in Windows). Specifies the text used for the **Open** button in the dialog box. If this argument is omitted, the button text is "Open."

MultiSelect Optional **Variant**. **True** to allow multiple file names to be selected. **False** to allow only one file name to be selected. The default value is **False**

Remarks

This method returns the selected file name or the name entered by the user. The returned name may include a path specification. If ***MultiSelect*** is **True**, the return value is an array of the selected file names (even if only one filename is selected). Returns **False** if the user cancels the dialog box.

This method may change the current drive or folder.

GetOpenFilename Method Example

This example displays the **Open** dialog box, with the file filter set to text files. If the user chooses a file name, the code displays that file name in a message box.

```
fileToOpen = Application.GetOpenFilename("Text Files (*.txt), *.txt")
If fileToOpen <> False Then
   MsgBox "Open " & fileToOpen
End If
```

This is the same example in Microsoft Excel for the Macintosh.

```
fileToOpen = Application.GetOpenFilename("TEXT")
If fileToOpen <> False Then
   MsgBox "Open " & fileToOpen
End If
```

GetSaveAsFilename Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGetSaveAsFilenameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGetSaveAsFilenameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGetSaveAsFilenameA "}

Displays the standard **Save As** dialog box and gets a file name from the user without actually saving any files.

Syntax

expression.**GetSaveAsFilename(*InitialFilename, FileFilter, FilterIndex, Title, ButtonText*)**

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

InitialFilename Optional **Variant**. Specifies the suggested file name. If this argument is omitted, Microsoft Excel uses the active workbook's name.

FileFilter Optional **Variant**. A string specifying file filtering criteria.

In Windows, this string consists of pairs of file filter strings followed by the MS-DOS wildcard file filter specification, with each part and each pair separated by commas. Each separate pair is listed in the **Files of type** drop-down list box. For example, the following string specifies two file filters, text and addin: "Text Files (*.txt), *.txt, Add-In Files (*.xla), *.xla".

To use multiple MS-DOS wildcard expressions for a single file filter type, separate the wildcard expressions with semicolons; for example, "Visual Basic Files (*.bas; *.txt),*.bas;*.txt".

If omitted in Windows, this argument defaults to "All Files (*.*),*.*".

On the Macintosh, this string is a list of comma-separated file type codes (for example, "TEXT,XLA5,XLS4"). Spaces are significant and shouldn't be inserted before or after the comma separators unless they're part of the file type code. If omitted, this argument defaults to all file types.

FilterIndex Optional **Variant**. Windows only (ignored on the Macintosh). Specifies the index number of the default file filtering criteria, from 1 to the number of filters specified in ***FileFilter***. If this argument is omitted or greater than the number of filters present, the first file filter is used.

Title Optional **Variant**. Specifies the title of the dialog box. If this argument is omitted, the default title is used.

ButtonText Optional **Variant**. Macintosh only (ignored in Windows). Specifies the text used for the **Save** button in the dialog box. If this argument is omitted, the button text is "Save".

Remarks

This method returns the selected file name or the name entered by the user. The returned name may include a path specification. Returns **False** if the user cancels the dialog box.

This method may change the current drive or folder.

GetSaveAsFilename Method Example

This example displays the **Save As** dialog box, with the file filter set to text files. If the user chooses a file name, the example displays that file name in a message box.

```
fileSaveName = Application.GetSaveAsFilename( _
   fileFilter:="Text Files (*.txt), *.txt")
If fileSaveName <> False Then
   MsgBox "Save as " & fileSaveName
End If
```

This is the same example in Microsoft Excel for the Macintosh.

```
fileSaveName = Application.GetSaveAsFilename( _
   fileFilter:="TEXT")
If fileSaveName <> False Then
   MsgBox "Save as " & FileSaveName
End If
```

GridlineColorIndex Property

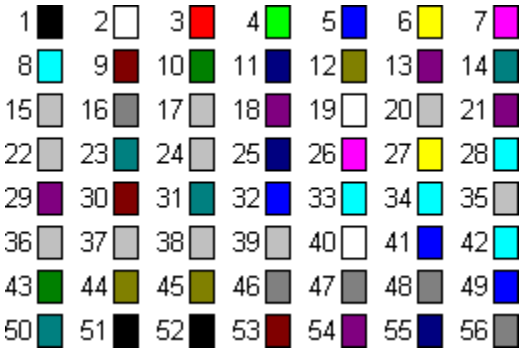
{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGridlineColorIndexC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGridlineColorIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGridlineColorIndexA "}

Returns or sets the gridline color as an index into the current color palette, or as one of the following **XlColorIndex** constants: **xlColorIndexAutomatic** or **xlColorIndexNone**. Read/write **Variant**.

Remarks

Set this property to **xlColorIndexAutomatic** to specify the automatic color.

The following illustration shows the color-index values in the default color palette.



GridlineColorIndex Property Example

This example sets the gridline color in the active window to blue.

```
ActiveWindow.GridlineColorIndex = 5
```

HasDataLabels Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasDataLabelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasDataLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasDataLabelsA "}

True if the series has data labels. Read/write **Boolean**.

HasDataLabels Property Example

This example turns on data labels for series three in Chart1.

```
With Charts("Chart1").SeriesCollection(3)
    .HasDataLabels = True
    .ApplyDataLabels type:=xlValue
End With
```

HasDropLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasDropLinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasDropLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasDropLinesA "}

True if the line chart or area chart has drop lines. Applies only to line and area charts. Read/write **Boolean**.

HasDropLines Property Example

This example turns on drop lines for chart group one in Chart1 and then sets their line style, weight, and color. The example should be run on a 2-D line chart that has one series.

```
With Charts("Chart1").ChartGroups(1)
   .HasDropLines = True
   With .DropLines.Border
      .LineStyle = xlThin
      .Weight = xlMedium
      .ColorIndex = 3
   End With
End With
```

HasHiLoLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasHiLoLinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasHiLoLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasHiLoLinesA "}

True if the line chart has high-low lines. Applies only to line charts. Read/write **Boolean**.

HasHiLoLines Property Example

This example turns on high-low lines for chart group one in Chart1 and then sets line style, weight, and color. The example should be run on a 2-D line chart that has three series of stock-quote-like data (high-low-close).

```
With Charts("Chart1").ChartGroups(1)
    .HasHiLoLines = True
    With .HiLoLines.Border
        .LineStyle = xlThin
        .Weight = xlMedium
        .ColorIndex = 3
    End With
End With
```

HasRadarAxisLabels Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasRadarAxisLabelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasRadarAxisLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasRadarAxisLabelsA "}

True if a radar chart has axis labels. Applies only to radar charts. Read/write **Boolean**.

HasRadarAxisLabels Property Example

This example turns on radar axis labels for chart group one in Chart1 and sets their color. The example should be run on a radar chart.

```
With Charts("Chart1").ChartGroups(1)
    .HasRadarAxisLabels = True
    .RadarAxisLabels.Font.ColorIndex = 3
End With
```

HasSeriesLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasSeriesLinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasSeriesLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasSeriesLinesA "}

True if a stacked column chart or bar chart has series lines or if a Pie of Pie chart or Bar of Pie chart has connector lines between the two sections. Applies only to stacked column charts, bar charts, Pie of Pie charts, or Bar of Pie charts. Read/write **Booean**.

HasSeriesLines Property Example

This example turns on series lines for chart group one in Chart1 and then sets their line style, weight, and color. The example should be run on a 2-D stacked column chart that has two or more series.

```
With Charts("Chart1").ChartGroups(1)
   .HasSeriesLines = True
   With .SeriesLines.Border
      .LineStyle = xlThin
      .Weight = xlMedium
      .ColorIndex = 3
   End With
End With
```

HasUpDownBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasUpDownBarsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasUpDownBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasUpDownBarsA "}

True if a line chart has up and down bars. Applies only to line charts. Read/write **Boolean**

HasUpDownBars Property Example

This example turns on up and down bars for chart group one in Chart1 and then sets their colors. The example should be run on a 2-D line chart containing two series that cross each other at one or more data points.

```
With Charts("Chart1").ChartGroups(1)
    .HasUpDownBars = True
    .DownBars.Interior.ColorIndex = 3
    .UpBars.Interior.ColorIndex = 5
End With
```

HiLoLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHiLoLinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHiLoLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHiLoLinesA "}

Returns a **HiLoLines** object that represents the high-low lines for a series on a line chart. Applies only to line charts. Read-only.

HiLoLines Property Example

This example turns on high-low lines for chart group one in Chart1 and then sets their line style, weight, and color. The example should be run on a 2-D line chart that has three series of stock-quote-like data (high-low-close).

```
With Charts("Chart1").ChartGroups(1)
    .HasHiLoLines = True
    With .HiLoLines.Border
        .LineStyle = xlThin
        .Weight = xlMedium
        .ColorIndex = 3
    End With
End With
```

InchesToPoints Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthInchesToPointsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthInchesToPointsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthInchesToPointsA "}

Converts a measurement from inches to points.

Syntax

expression.**InchesToPoints(*Inches*)**

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

Inches Required **Double**. Specifies the inch value to be converted to points.

InchesToPoints Method Example

This example sets the left margin of Sheet1 to 2.5 inches.

```
Worksheets("Sheet1").PageSetup.LeftMargin = _
      Application.InchesToPoints(2.5)
```

Italic Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItalicC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItalicX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItalicA "}

True if the font style is italic. Read/write **Boolean**.

Italic Property Example

This example sets the font style to italic for the range A1:A5 on Sheet1.

```
Worksheets("Sheet1").Range("A1:A5").Font.Italic = True
```

LargeScroll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthLargeScrollC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthLargeScrollX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthLargeScrollA "}

Scrolls the contents of the window by pages.

Syntax

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

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

Down Optional **Variant**. The number of pages to scroll the contents down.

Up Optional **Variant**. The number of pages to scroll the contents up.

ToRight Optional **Variant**. The number of pages to scroll the contents to the right.

ToLeft Optional **Variant**. The number of pages to scroll the contents to the left.

Remarks

If ***Down*** and ***Up*** are both specified, the contents of the window are scrolled by the difference of the arguments. For example, if ***Down*** is 3 and ***Up*** is 6, the contents are scrolled up three pages.

If ***ToLeft*** and ***ToRight*** are both specified, the contents of the window are scrolled by the difference of the arguments. For example, if ***ToLeft*** is 3 and ***ToRight*** is 6, the contents are scrolled to the right three pages.

Any of the arguments can be a negative number.

LargeScroll Method Example

This example scrolls the contents of the active window of Sheet1 down three pages.

```
Worksheets("Sheet1").Activate
ActiveWindow.LargeScroll down:=3
```

Outline Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOutlineC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOutlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOutlineA "}

Returns an **Outline** object that represents the outline for the specified worksheet. Read-only.

Outline Property Example

This example sets the outline on Sheet1 to use automatic styles.

```
Worksheets("Sheet1").Outline.AutomaticStyles = True
```

Overlap Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOverlapC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOverlapX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOverlapA "}

Specifies how bars and columns are positioned. Can be a value between – 100 and 100. Applies only to 2-D bar and 2-D column charts. Read/write **Long**.

Remarks

If this property is set to – 100, bars are positioned so that there's one bar width between them. If the overlap is 0 (zero), there's no space between bars (one bar starts immediately after the preceding bar). If the overlap is 100, bars are positioned on top of each other.

Overlap Property Example

This example sets the overlap for chart group one to – 50. The example should be run on a 2-D column chart that has two or more series.

```
Charts("Chart1").ChartGroups(1).Overlap = -50
```

RadarAxisLabels Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRadarAxisLabelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRadarAxisLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRadarAxisLabelsA "}

Returns a **TickLabels** object that represents the radar axis labels for the specified chart group. Read-only.

RadarAxisLabels Property Example

This example turns on radar axis labels for chart group one in Chart1 and then sets the color for the labels. The example should be run on a radar chart.

```
With Charts("Chart1").ChartGroups(1)
    .HasRadarAxisLabels = True
    .RadarAxisLabels.Font.ColorIndex = 3
End With
```

RefersToLocal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefersToLocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefersToLocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefersToLocalA "}

Returns or sets the formula that the name refers to. The formula is in the language of the user, and it's in A1-style notation, beginning with an equal sign. Read/write **String**.

RefersToLocal Property Example

This example creates a new worksheet and then inserts a list of all the names in the active workbook, including their formulas (in A1-style notation and in the language of the user).

```
Set newSheet = ActiveWorkbook.Worksheets.Add
i = 1
For Each nm In ActiveWorkbook.Names
   newSheet.Cells(i, 1).Value = nm.NameLocal
   newSheet.Cells(i, 2).Value = "'" & nm.RefersToLocal
   i = i + 1
Next
```

RefersToR1C1 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefersToR1C1C "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefersToR1C1X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefersToR1C1A "}

Returns or sets the formula that the name refers to. The formula is in the language of the macro, and it's in R1C1-style notation, beginning with an equal sign. Read/write **String**.

RefersToR1C1 Property Example

This example creates a new worksheet and then inserts a list of all the names in the active workbook, including their formulas (in R1C1-style notation and in the language of the macro).

```
Set newSheet = ActiveWorkbook.Worksheets.Add
i = 1
For Each nm In ActiveWorkbook.Names
   newSheet.Cells(i, 1).Value = nm.Name
   newSheet.Cells(i, 2).Value = "'" & nm.RefersToR1C1
   i = i + 1
Next
```

RefersToR1C1Local Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefersToR1C1LocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefersToR1C1LocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefersToR1C1LocalA "}

Returns or sets the formula that the name refers to. This formula is in the language of the user, and it's in R1C1-style notation, beginning with an equal sign. Read/write **String**.

RefersToR1C1Local Property Example

This example creates a new worksheet and then inserts a list of all the names in the active workbook, including their formulas (in R1C1-style notation and in the language of the user).

```
Set newSheet = ActiveWorkbook.Worksheets.Add
i = 1
For Each nm In ActiveWorkbook.Names
   newSheet.Cells(i, 1).Value = nm.NameLocal
   newSheet.Cells(i, 2).Value = "'" & nm.RefersToR1C1Local
   i = i + 1
Next
```

SeriesLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSeriesLinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSeriesLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSeriesLinesA "}

Returns a **SeriesLines** object that represents the series lines for a stacked bar chart or a stacked column chart. Applies only to stacked bar and stacked column charts. Read-only.

SeriesLines Property Example

This example turns on series lines for chart group one in Chart1 and then sets their line style, weight, and color. The example should be run on a 2-D stacked column chart that has two or more series.

```
With Charts("Chart1").ChartGroups(1)
   .HasSeriesLines = True
   With .SeriesLines.Border
      .LineStyle = xlThin
      .Weight = xlMedium
      .ColorIndex = 3
   End With
End With
```

SmallScroll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSmallScrollC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSmallScrollX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSmallScrollA "}

Scrolls the contents of the window by rows or columns.

Syntax

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

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

Down Optional **Variant**. The number of rows to scroll the contents down.

Up Optional **Variant**. The number of rows to scroll the contents up.

ToRight Optional **Variant**. The number of columns to scroll the contents to the right.

ToLeft Optional **Variant**. The number of columns to scroll the contents to the left.

Remarks

If ***Down*** and ***Up*** are both specified, the contents of the window are scrolled by the difference of the arguments. For example, if ***Down*** is 3 and ***Up*** is 6, the the contents are scrolled up three rows.

If ***ToLeft*** and ***ToRight*** are both specified, the contents of the window are scrolled by the difference of the arguments. For example, if ***ToLeft*** is 3 and ***ToRight*** is 6, the contents are scrolled to the right three columns.

Any of these arguments can be a negative number.

SmallScroll Method Example

This example scrolls the contents of the active window of Sheet1 down three rows.

```
Worksheets("Sheet1").Activate
ActiveWindow.SmallScroll down:=3
```

SortSpecial Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSortSpecialC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSortSpecialX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSortSpecialA "}

Syntax 1: Uses Far East sorting methods to sort the range, or uses the current region if the range contains only one cell.

Syntax 2: Uses Far East sorting methods to sort a PivotTable. For more information, see the argument list.

Syntax 1

expression.**SortSpecial(*SortMethod, Key1, Order1, Key2, Type, Order2, Key3, Order3, Header, OrderCustom, MatchCase, Orientation*)**

Syntax 2

expression.**SortSpecial(*SortMethod, Key1, Order1, Type, OrderCustom, Orientation*)**

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

SortMethod Optional **Variant**. Specifies how to sort. Can be one of the following **XlSortMethod** constants: **xlSyllabary** (to sort phonetically) or **xlCodePage** (to sort by code page). The default value is **xlSyllabary**.

Key1 Optional **Variant**. The first sort field, as either text (a pivot field or range name) or a **Range** object ("Dept" or `Cells(1, 1)`, for example).

Order1 Optional **Variant**. Can be one of the following **XlSortOrder** constants: **xlAscending** or **xlDescending**. Use **xlAscending** to sort ***Key1*** in ascending order. Use **xlDescending** to sort ***Key1*** in descending order. The default value is **xlAscending**.

Key2 Optional **Variant**. The second sort field, as either text (a pivot field or range name) or a **Range** object. If this argument is omitted, there's no second sort field. Not used when sorting PivotTables.

Type Optional **Variant**. Specifies which elements are sorted. Can be one of the following **XlSortType** constants: **xlSortValues** or **xlSortLabels**. Used only when sorting PivotTables.

Order2 Optional **Variant**. Can be one of the following **XlSortOrder** constants: **xlAscending** or **xlDescending**. Use **xlAscending** to sort ***Key2*** in ascending order. Use **xlDescending** to sort ***Key2*** in descending order. The default value is **xlAscending**. Not used when sorting PivoTables.

Key3 Optional **Variant**. The third sort field, as either text (a range name) or a **Range** object. If this argument is omitted, there's no third sort field. Not used when sorting PivotTables.

Order3 Optional **Variant**. Can be one of the following **XlSortOrder** constants: **xlAscending** or **xlDescending**. Use **xlAscending** to sort ***Key3*** in ascending order. Use **xlDescending** to sort ***Key3*** in descending order. The default value is **xlAscending**. Not used when sorting PivotTables.

Header Optional **Variant**. Specifies whether the first row contains headers. Can be one of the following **XlYesNoGuess** constants: **xlYes**, **xlNo**, or **xlGuess**. Use **xlYes** if the first row contains headers (it shouldn't be sorted). Use **xlNo** if there are no headers (the entire range should be sorted). Use **xlGuess** to let Microsoft Excel determine whether there's a header, and to determine where it is, if there is one. The default value is **xlNo**. Not used when sorting PivotTables.

OrderCustom Optional **Variant**. 1-based integer offset into the list of custom sort orders. If this argument is omitted, 1 (Normal) is used.

MatchCase Optional **Variant**. **True** to do a case-sensitive sort; **False** to do a sort that's not case sensitive. Not used when sorting PivotTables.

Orientation Optional **Variant**. If **xlTopToBottom** or omitted, the sort is done from top to bottom (by row). If **xlLeftToRight**, the sort is done from left to right (by column).

SortSpecial Method Example

This example sorts the range A1:G37 on Sheet1, using cell A1 as the first sort key and cell C1 as the second sort key. The sort is done in ascending code page order by row, and there are no headers.

```
Worksheets("Sheet1").Range("A1:G37").SortSpecial _
   sortMethod:=xlCodePage, _
   key1:=Range("A1"), order1:=xlAscending, _
   key2:=Range("C1"), order2:=xlAscending
```

SourceData Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSourceDataC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSourceDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSourceDataA "}

Returns the data source for the PivotTable, as shown in the following table. Read-only **Variant**.

Data source	**Return value**
Microsoft Excel list or database	The cell reference, as text.
External data source	An array. Each row consists of an SQL connection string with the remaining elements as the query string, broken down into 200-character segments.
Multiple consolidation ranges	A two-dimensional array. Each row consists of a reference and its associated page field items.
Another PivotTable	One of the above three kinds of information.

SourceData Property Example

Assume that you used an external data source to create a PivotTable on Sheet1. This example inserts the SQL connection string and query string into a new worksheet.

```
Set newSheet = ActiveWorkbook.Worksheets.Add
sdArray = Worksheets("Sheet1").UsedRange.PivotTable.SourceData
For i = LBound(sdArray) To UBound(sdArray)
   newSheet.Cells(i, 1) = sdArray(i)
Next i
```

Strikethrough Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStrikethroughC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStrikethroughX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStrikethroughA "}

True if the font is struck through with a horizontal line. Read/write **Boolean**.

Strikethrough Property Example

This example sets the font in the active cell on Sheet1 to strikethrough.

```
Worksheets("Sheet1").Activate
ActiveCell.Font.Strikethrough = True
```

Underline Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUnderlineC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUnderlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUnderlineA "}

Returns or sets the type of underline applied to the font. Can be one of the following **XlUnderlineStyle** constants: **xlUnderlineStyleNone**, **xlUnderlineStyleSingle**, **xlUnderlineStyleDouble**, **xlUnderlineStyleSingleAccounting**, or **xlUnderlineStyleDoubleAccounting**. Read/write **Long**.

Underline Property Example

This example sets the font in the active cell on Sheet1 to single underline.

```
Worksheets("Sheet1").Activate
ActiveCell.Font.Underline = xlUnderlineStyleSingle
```

UpBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUpBarsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUpBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUpBarsA "}

Returns an **UpBars** object that represents the up bars on a line chart. Applies only to line charts. Read-only.

UpBars Property Example

This example turns on up and down bars for chart group one in Chart1 and then sets their colors. The example should be run on a 2-D line chart containing two series that cross each other at one or more data points.

```
With Charts("Chart1").ChartGroups(1)
    .HasUpDownBars = True
    .DownBars.Interior.ColorIndex = 3
    .UpBars.Interior.ColorIndex = 5
End With
```

VaryByCategories Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVaryByCategoriesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVaryByCategoriesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVaryByCategoriesA "}

True if Microsoft Excel assigns a different color or pattern to each data marker. The chart must contain only one series. Read/write **Boolean**.

VaryByCategories Property Example

This example assigns a different color or pattern to each data marker in chart group one. The example should be run on a 2-D line chart that has data markers on a series.

```
Charts("Chart1").ChartGroups(1).VaryByCategories = True
```

ExclusiveAccess Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthExclusiveAccessC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthExclusiveAccessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthExclusiveAccessA "}

Assigns the current user exclusive access to the workbook that's open as a shared list.

Syntax

expression.**ExclusiveAccess**

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

Remarks

The **ExclusiveAccess** method saves any changes you've made to the workbook and requires other users who have the workbook open to save their changes to a different file.

If the specified workbook isn't open as a shared list, this method fails. To determine whether a workbook is open as a shared list, use the **MultiUserEditing** property.

ExclusiveAccess Method Example

This example determines whether the active workbook is open as a shared list. If it is, the example gives the current user exclusive access.

```
If ActiveWorkbook.MultiUserEditing Then
   ActiveWorkbook.ExclusiveAccess
End If
```

MultiUserEditing Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMultiUserEditingC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMultiUserEditingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMultiUserEditingA "}

True if the workbook is open as a shared list. Read-only **Boolean**.

Remarks

To save a workbook as a shared list, use the **SaveAs** method. To switch the workbook from shared mode to exclusive mode, use the **ExclusiveAccess** method.

MultiUserEditing Property Example

This example determines whether the active workbook is open in exclusive mode. If it is, the example saves the workbook as a shared list.

```
If Not ActiveWorkbook.MultiUserEditing Then
   ActiveWorkbook.SaveAs fileName:=ActiveWorkbook.FullName, _
      accessMode:=xlShared
End If
```

RevisionNumber Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRevisionNumberC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRevisionNumberX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRevisionNumberA "}

Returns the number of times the workbook has been saved while open as a shared list. If the workbook is open in exclusive mode, this property returns 0 (zero). Read-only **Long**.

Remarks

The **RevisionNumber** property is updated only when the local copy of the workbook is saved, not when remote copies are saved.

RevisionNumber Property Example

This example uses the revision number to determine whether the active workbook is open in exclusive mode. If it is, the example saves the workbook as a shared list.

```
If ActiveWorkbook.RevisionNumber = 0 Then
    ActiveWorkbook.SaveAs filename:=ActiveWorkbook.FullName, _
        accessMode:=xlShared, conflictResolution:=xlOtherSessionChanges
End If
```

ShowConflictHistory Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowConflictHistoryC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowConflictHistoryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowConflictHistoryA "}

True if the Conflict History worksheet is visible in the workbook that's open as a shared list. Read/write **Boolean**.

Remarks

If the specified workbook isn't open as a shared list, this property fails. To determine whether a workbook is open as a shared list, use the **MultiUserEditing** property.

ShowConflictHistory Property Example

This example determines whether the active workbook is open as a shared list. If it is, the example displays the Conflict History worksheet.

```
If ActiveWorkbook.MultiUserEditing Then
   ActiveWorkbook.ShowConflictHistory = True
End If
```

UserStatus Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUserStatusC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUserStatusX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUserStatusA "}

Returns a 1-based, two-dimensional array that provides information about each user who has the workbook open as a shared list. The first element of the second dimension is the name of the user, the second element is the date and time when the user last opened the workbook, and the third element is a number indicating the type of list (1 indicates exclusive, and 2 indicates shared). Read-only **Variant**.

Remarks

The **UserStatus** property doesn't return information about users who have the specified workbook open as read-only.

UserStatus Property Example

This example creates a new workbook and inserts into it information about all users who have the active workbook open as a shared list.

```
users = ActiveWorkbook.UserStatus
With Workbooks.Add.Sheets(1)
   For row = 1 To UBound(users, 1)
      .Cells(row, 1) = users(row, 1)
      .Cells(row, 2) = users(row, 2)
      Select Case users(row, 3)
         Case 1
            .Cells(row, 3).Value = "Exclusive"
         Case 2
            .Cells(row, 3).Value = "Shared"
      End Select
   Next
End With
```

ActivateMicrosoftApp Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthActivateMicrosoftAppC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthActivateMicrosoftAppX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthActivateMicrosoftAppA "}

Activates a Microsoft application. If the application is already running, this method activates the running application. If the application isn't running, this method starts a new instance of the application.

Syntax

expression.**ActivateMicrosoftApp(*index*)**

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

index Required **Long**. Specifies the Microsoft application to activate. Can be one of the following **XlMSApplication** constants: **xlMicrosoftWord**, **xlMicrosoftPowerPoint**, **xlMicrosoftMail**, **xlMicrosoftAccess**, **xlMicrosoftFoxPro**, **xlMicrosoftProject**, or **xlMicrosoftSchedulePlus**.

ActivateMicrosoftApp Method Example

This example starts and activates Word.

```
Application.ActivateMicrosoftApp xlMicrosoftWord
```

ActivateNext Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthActivateNextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthActivateNextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthActivateNextA "}

Activates the specified window and then sends it to the back of the window z-order.

Syntax

expression.**ActivateNext**

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

ActivateNext Method Example

This example sends the active window to the back of the z-order.

```
ActiveWindow.ActivateNext
```

ActivatePrevious Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthActivatePreviousC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthActivatePreviousX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthActivatePreviousA "}

Activates the specified window and then activates the window at the back of the window z-order.

Syntax

expression.**ActivatePrevious**

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

ActivatePrevious Method Example

This example activates the window at the back of the z-order.

```
ActiveWindow.ActivatePrevious
```

AxisTitle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAxisTitleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAxisTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAxisTitleA "}

Returns an **AxisTitle** object that represents the title of the specified axis. Read-only.

AxisTitle Property Example

This example adds an axis label to the category axis in Chart1.

```
With Charts("Chart1").Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Text = "July Sales"
End With
```

BCCRecipients Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBCCRecipientsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBCCRecipientsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBCCRecipientsA "}

Returns or sets the recipients of a blind carbon copy of the mailer. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read/write **Variant**.

Remarks

This property uses an array of strings specifying the address of each recipient, in one of the following formats:

- A record in the Preferred Personal Catalog. These names are one level deep ("Fred" or "June," for example).
- A full path specifying either a record in a personal catalog ("HD:Excel Folder:My Catalog:Barney") or a plain record ("HD:Folder:Martin").
- A relative path from the current working folder specifying either a personal catalog record ("My Catalog:Barney") or a plain record ("Martin").
- A path in a PowerShare catalog tree, in the form "CATALOG_NAME:<node>:RECORD_NAME", where <node> is a path to a PowerShare catalog. An example of a complete path is "AppleTalk:North Building Zone:George's Mac".

BCCRecipients Property Example

This example sets up the **Mailer** object for workbook one and then sends the workbook.

```
With Workbooks(1)
    .HasMailer = True
    With .Mailer
        .Subject = "Here is the workbook"
        .ToRecipients = Array("Jean")
        .CCRecipients = Array("Adam", "Bernard")
        .BCCRecipients = Array("Chris")
        .Enclosures = Array("TestFile")
    End With
    .SendMailer
End With
```

CategoryNames Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCategoryNamesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCategoryNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCategoryNamesA "}

Returns or sets all the category names for the specified axis, as a text array. When you set this property, you can set it to either an array or a **Range** object that contains the category names. Read/write **Variant**.

Remarks

Category names are really a property of the "special" series in an axis grouping. Deleting or modifying that special series will change the category names for all series using the axis.

CategoryNames Property Example

This example sets the category names for Chart1 to the values in cells B1:B5 on Sheet1.

```
Set Charts("Chart1").Axes(xlCategory).CategoryNames = _
   Worksheets("Sheet1").Range("B1:B5")
```

This example uses an array to set individual category names for Chart1.

```
Charts("Chart1").Axes(xlCategory).CategoryNames = _
   Array ("1985", "1986", "1987", "1988", "1989")
```

CCRecipients Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCCRecipientsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCCRecipientsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCCRecipientsA "}

Returns or sets the recipients of a carbon copy (an indirect copy) of the mailer. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read/write **Variant**.

Remarks

This property uses an array of strings specifying the address, in one of the following formats:

- A record in the Preferred Personal Catalog. These names are one level deep ("Fred" or "June," for example).
- A full path specifying either a record in a personal catalog ("HD:Excel Folder:My Catalog:Barney") or a plain record ("HD:Folder:Martin").
- A relative path from the current working folder specifying either a personal catalog record ("My Catalog:Barney") or a plain record ("Martin").
- A path in a PowerShare catalog tree, in the form "CATALOG_NAME:<node>:RECORD_NAME", where <node> is a path to a PowerShare catalog. An example of a complete path is "AppleTalk:North Building Zone:George's Mac".

CCRecipients Property Example

This example sets up the **Mailer** object for workbook one and then sends the workbook.

```
With Workbooks(1)
    .HasMailer = True
    With .Mailer
        .Subject = "Here is the workbook"
        .ToRecipients = Array("Jean")
        .CCRecipients = Array("Adam", "Bernard")
        .BCCRecipients = Array("Chris")
        .Enclosures = Array("TestFile")
    End With
    .SendMailer
End With
```

ChartTitle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChartTitleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChartTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChartTitleA "}

Returns a **ChartTitle** object that represents the title of the specified chart. Read-only.

ChartTitle Property Example

This example sets the text for the title of Chart1.

```
With Charts("Chart1")
    .HasTitle = True
    .ChartTitle.Text = "First Quarter Sales"
End With
```

Deselect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDeselectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDeselectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDeselectA "}

Cancels the selection for the specified chart.

Syntax

expression.**Deselect**

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

Deselect Method Example

This example is equivalent to pressing ESC while working on the active chart. The example should be run on a chart that has a component (such as an axis) selected.

```
ActiveChart.Deselect
```

DoughnutHoleSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDoughnutHoleSizeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDoughnutHoleSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDoughnutHoleSizeA "}

Returns or sets the size of the hole in a doughnut chart group. The hole size is expressed as a percentage of the chart size, between 10 and 90 percent. Read/write **Long**.

DoughnutHoleSize Property Example

This example sets the hole size for doughnut group one in Chart1. The example should be run on a 2-D doughnut chart.

```
Charts("Chart1").DoughnutGroups(1).DoughnutHoleSize = 10
```

EnableCancelKey Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableCancelKeyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableCancelKeyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableCancelKeyA "}

Controls how Microsoft Excel handles CTRL+BREAK (or ESC or COMMAND+PERIOD) user interruptions to the running procedure. Read/write **Long**.

Can be one of the following **XlEnableCancelKey** constants.

Constant	Meaning
xlDisabled	Cancel key trapping is completely disabled.
xlInterrupt	The current procedure is interrupted, and the user can debug or end the procedure.
xlErrorHandler	The interrupt is sent to the running procedure as an error, trappable by an error handler set up with an **On Error GoTo** statement. The trappable error code is 18.

Remarks

Use this property very carefully. If you use **xlDisabled**, there's no way to interrupt a runaway loop or other non – self-terminating code. Likewise, if you use **xlErrorHandler** but your error handler always returns using the **Resume** statement, there's no way to stop runaway code.

The **EnableCancelKey** property is always reset to **xlInterrupt** whenever Microsoft Excel returns to the idle state and there's no code running. To trap or disable cancellation in your procedure, you must explicitly change the **EnableCancelKey** property every time the procedure is called.

EnableCancelKey Property Example

This example shows how you can use the **EnableCancelKey** property to set up a custom cancellation handler.

```
On Error GoTo handleCancel
Application.EnableCancelKey = xlErrorHandler
MsgBox "This may take a long time: press ESC to cancel"
For x = 1 To 1000000 ' Do something 1,000,000 times (long!)
   ' do something here
Next x

handleCancel:
If Err = 18 Then
   MsgBox "You cancelled"
End If
```

Enclosures Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnclosuresC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnclosuresX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnclosuresA "}

Returns or sets the enclosed files that are attached to the workbook mailer, as an array of strings, with each string indicating the path of a file to attach as an enclosure. Relative paths are allowed; they're assumed to be based on the current folder. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read/write **Variant**.

Enclosures Property Example

This example sets up the **Mailer** object for workbook one and then sends the workbook.

```
With Workbooks(1)
    .HasMailer = True
    With .Mailer
        .Subject = "Here is the workbook"
        .ToRecipients = Array("Jean")
        .CCRecipients = Array("Adam", "Bernard")
        .BCCRecipients = Array("Chris")
        .Enclosures = Array("TestFile")
    End With
    .SendMailer
End With
```

EndStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEndStyleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndStyleA "}

Returns or sets the end style for the error bars. Can be one of the following **XlEndStyleCap** constants: **xlCap** or **xlNoCap**. Read/write **Long**.

EndStyle Property Example

This example sets the end style for the error bars for series one in Chart1. The example should be run on a 2-D line chart that has Y error bars for the first series.

```
Charts("Chart1").SeriesCollection(1).ErrorBars.EndStyle = xlCap
```

ForwardMailer Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthForwardMailerC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthForwardMailerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthForwardMailerA "}

Sets up the workbook mailer for forwarding by creating a new mailer that is preset with the subject and enclosures of the existing mailer. Valid only when the workbook has a received mailer attached (you can only forward a workbook you've received). Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed.

Syntax

expression.**ForwardMailer**

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

Remarks

After you use this method to set up a workbook mailer for forwarding, you can change the mailer settings (if necessary) by using the **Mailer** property and then use the **SendMailer** method to forward the workbook.

This method generates an error if it's used in Microsoft Windows.

ForwardMailer Method Example

This example forwards the active workbook.

```
ActiveWorkbook.ForwardMailer
```

HasMailer Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasMailerC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasMailerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasMailerA "}

True if the workbook has a mailer. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read/write **Boolean**.

HasMailer Property Example

This example replies to the sender of the active workbook.

```
Set original = ActiveWorkbook
If original.HasMailer Then
   original.Reply
   original.Mailer.Subject = "Here's my reply"
   ActiveWorkbook.SendMailer
End If
```

HasRoutingSlip Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasRoutingSlipC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasRoutingSlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasRoutingSlipA "}

True if the workbook has a routing slip. Read/write **Boolean**.

Remarks

Setting this property to **True** creates a routing slip with default values. Setting the property to **False** deletes the routing slip.

HasRoutingSlip Property Example

This example creates a routing slip for Book1.xls and then sends the workbook to three recipients, one after another.

```
Workbooks("BOOK1.XLS").HasRoutingSlip = True
With Workbooks("BOOK1.XLS").RoutingSlip
    .Delivery = xlOneAfterAnother
    .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
    .Subject = "Here is BOOK1.XLS"
    .Message = "Here is the workbook. What do you think?"
End With
Workbooks("BOOK1.XLS").Route
```

Mailer Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMailerC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMailerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMailerA "}

Returns a **Mailer** object that represents the PowerTalk mailer attached to the workbook. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read-only.

Remarks

The **Mailer** object contains the properties needed to mail workbooks with PowerTalk. To mail a workbook, turn on the mailer by using the **HasMailer** property, set the mailer properties, and then send the workbook and mailer by using the **SendMailer** method.

Mailer Property Example

This example sets up the **Mailer** object for workbook one and then sends the workbook.

```
With Workbooks(1)
    .HasMailer = True
    With .Mailer
        .Subject = "Here is the workbook"
        .ToRecipients = Array("Jean")
        .CCRecipients = Array("Adam", "Bernard")
        .BCCRecipients = Array("Chris")
        .Enclosures = Array("TestFile")
    End With
    .SendMailer
End With
```

MarkerBackgroundColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarkerBackgroundColorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarkerBackgroundColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarkerBackgroundColorA "}

Returns or sets the marker background color as an RGB value. Applies only to line, scatter, and radar charts. Read/write **Long**

MarkerBackgroundColor Property Example

This example sets the marker background and foreground colors for the second point in series one in Chart1.

```
With Charts("Chart1").SeriesCollection(1).Points(2)
    .MarkerBackgroundColor = RGB(0,255,0) ' green
    .MarkerForegroundColor = RGB(255,0,0) ' red
End With
```

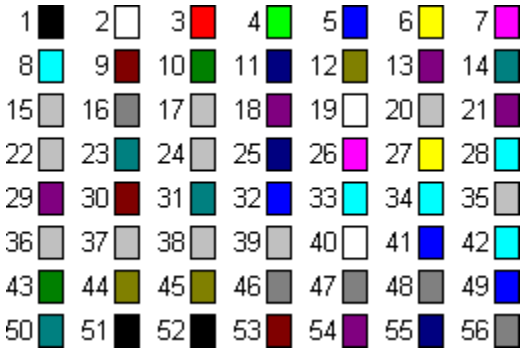
MarkerBackgroundColorIndex Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarkerBackgroundColorIndexC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarkerBackgroundColorIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarkerBackgroundColorIndexA "}

Returns or sets the marker background color as an index into the current color palette, or as one of the following **XlColorIndex** constants: **xlColorIndexAutomatic** or **xlColorIndexNone**. Applies only to line, scatter, and radar charts. Read/write **Long**.

Remarks

The following illustration shows the color-index values in the default color palette.



MarkerBackgroundColorIndex Property Example

This example sets the marker background and foreground colors for the second point in series one in Chart1.

```
With Charts("Chart1").SeriesCollection(1).Points(2)
    .MarkerBackgroundColorIndex = 4 'green
    .MarkerForegroundColorIndex = 3 'red
End With
```

MarkerForegroundColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarkerForegroundColorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarkerForegroundColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarkerForegroundColorA "}

Returns or sets the foreground color of the marker as an RGB value. Applies only to line, scatter, and radar charts. Read/write **Long**.

MarkerForegroundColor Property Example

This example sets the marker background and foreground colors for the second point in series one in Chart1.

```
With Charts("Chart1").SeriesCollection(1).Points(2)
    .MarkerBackgroundColor = RGB(0,255,0) ' green
    .MarkerForegroundColor = RGB(255,0,0) ' red
End With
```

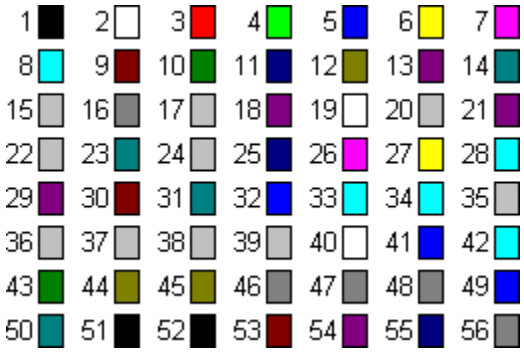
MarkerForegroundColorIndex Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarkerForegroundColorIndexC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarkerForegroundColorIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarkerForegroundColorIndexA "}

Returns or sets the marker foreground color as an index into the current color palette, or as one of the following **XlColorIndex** constants: **xlColorIndexAutomatic** or **xlColorIndexNone**. Applies only to line, scatter, and radar charts. Read/write **Long**.

Remarks

The following illustration shows the color-index values in the default color palette.



MarkerForegroundColorIndex Property Example

This example sets the marker background and foreground colors for the second point in series one in Chart1.

```
With Charts("Chart1").SeriesCollection(1).Points(2)
    .MarkerBackgroundColorIndex = 4 'green
    .MarkerForegroundColorIndex = 3 'red
End With
```

NavigateArrow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthNavigateArrowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthNavigateArrowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthNavigateArrowA "}

Navigates a tracer arrow for the specified range to the precedent, dependent, or error-causing cell or cells. Selects the precedent, dependent, or error cells and returns a **Range** object that represents the new selection. This method causes an error if it's applied to a cell without visible tracer arrows.

Syntax

expression.**NavigateArrow(*TowardPrecedent, ArrowNumber, LinkNumber*)**

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

TowardPrecedent Optional **Variant**. Specifies the direction to navigate: **True** to navigate toward precedents, **False** to navigate toward dependent.

ArrowNumber Optional **Variant**. Specifies the arrow number to navigate; corresponds to the numbered reference in the cell's formula.

LinkNumber Optional **Variant**. If the arrow is an external reference arrow, this argument indicates which external reference to follow. If this argument is omitted, the first external reference is followed.

NavigateArrow Method Example

This example navigates along the first tracer arrow from cell A1 on Sheet1 toward the precedent cell. The example should be run on a worksheet containing a formula in cell A1 that includes references to cells D1, D2, and D3 (for example, the formula =D1*D2*D3). Before running the example, display the **Auditing** toolbar, select cell A1, and click the **Trace Precedents** button.

```
Worksheets("Sheet1").Activate
Range("A1").Select
ActiveCell.NavigateArrow True, 1
```

OpenText Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOpenTextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOpenTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOpenTextA "}

Loads and parses a text file as a new workbook with a single sheet that contains the parsed text-file data.

Syntax

expression.**OpenText(*Filename, Origin, StartRow, DataType, TextQualifier, ConsecutiveDelimiter, Tab, Semicolon, Comma, Space, Other, OtherChar, FieldInfo*)**

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

Filename Required **String**. Specifies the file name of the text file to be opened and parsed.

Origin Optional **Variant**. Specifies the origin of the text file. Can be one of the following **XlPlatform** constants: **xlMacintosh**, **xlWindows**, or **xlMSDOS**. If this argument is omitted, the method uses the current setting of the **File Origin** option in the **Text Import Wizard**.

StartRow Optional **Variant**. The row number at which to start parsing text. The default value is 1.

DataType Optional **Variant**. Specifies the column format of the data in the file. Can be one of the following **XlTextParsingType** constants: **xlDelimited** or **xlFixedWidth**. The default value is **xlDelimited**.

TextQualifier Optional **Variant**. Specifies the text qualifier. Can be one of the following **XlTextQualifier** constants: **xlTextQualifierDoubleQuote**, **xlTextQualifierSingleQuote**, or **xlTextQualifierNone**. The default value is **xlTextQualifierDoubleQuote**.

ConsecutiveDelimiter Optional **Variant**. **True** to have consecutive delimiters considered one delimiter. The default is **False**.

Tab Optional **Variant**. **True** to have the tab character be the delimiter (***DataType*** must be **xlDelimited**). The default value is **False**.

Semicolon Optional **Variant**. **True** to have the semicolon character be the delimiter (***DataType*** must be **xlDelimited**). The default value is **False**.

Comma Optional **Variant**. **True** to have the comma character be the delimiter (***DataType*** must be **xlDelimited**). The default value is **False**.

Space Optional **Variant**. **True** to have the space character be the delimiter (***DataType*** must be **xlDelimited**). The default value is **False**.

Other Optional **Variant**. **True** to have the character specified by the ***OtherChar*** argument be the delimiter (***DataType*** must be **xlDelimited**). The default value is **False**.

OtherChar Optional **Variant** (required if ***Other*** is **True**). Specifies the delimiter character when ***Other*** is **True**. If more than one character is specified, only the first character of the string is used; the remaining characters are ignored.

FieldInfo Optional **Variant**. An array containing parse information for individual columns of data. The interpretation depends on the value of ***DataType***.

When the data is delimited, this argument is an array of two-element arrays, with each two-element array specifying the conversion options for a particular column. The first element is the column number (1-based), and the second element is one of the following numbers, specifying how the column in parsed.

1	General
2	Text
3	MDY date
4	DMY date
5	YMD date
6	MYD date

7	DYM date
8	YDM date
9	Skip the column

The column specifiers can be in any order. If there's no column specifier for a particular column in the input data, the column is parsed with the General setting. This example causes the third column to be skipped, the first column to be parsed as text, and the remaining columns in the source data to be parsed with the General setting.

```
Array(Array(3, 9), Array(1, 2))
```

If the source data has fixed-width columns, the first element in each two-element array specifies the position of the starting character in the column (as an integer; character 0 (zero) is the first character). The second element in the two-element array specifies the parse option for the column as a number between 1 and 9, as listed in the preceding table.

The following example parses two columns from a fixed-width text file. The first column includes characters 1 through 10. Characters 11, 12, 13, 14, and 15 are skipped. The second column includes character 16 through the last character in the line.

```
Array(Array(0, 1), Array(10, 9), Array(15, 1))
```

OpenText Method Example

This example opens the file Data.txt and uses tab delimiters to parse the text file into a worksheet.

```
Workbooks.OpenText filename:="DATA.TXT", _
   dataType:=xlDelimited, tab:=True
```

OutlineFont Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOutlineFontC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOutlineFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOutlineFontA "}

True if the font is an outline font. Read/write **Boolean**.

Remarks

This property has no effect in Windows, but its value is retained (it can be set and returned).

OutlineFont Property Example

This example sets the font for cell A1 on Sheet1 to an outline font.

```
Worksheets("Sheet1").Range("A1").Font.OutlineFont = True
```

PatternColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPatternColorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPatternColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPatternColorA "}

Returns or sets the color of the interior pattern as an RGB value. Read/write **Variant**.

PatternColor Property Example

This example sets the color of the interior pattern for rectangle one on Sheet1.

```
With Worksheets("Sheet1").Rectangles(1).Interior
    .Pattern = xlGrid
    .PatternColor = RGB(255,0,0)
End With
```

PatternColorIndex Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPatternColorIndexC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPatternColorIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPatternColorIndexA "}

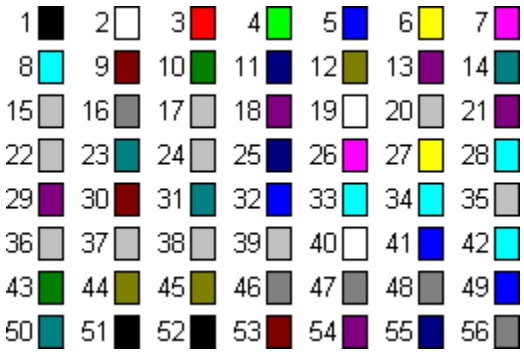
Returns or sets the color of the interior pattern as an index into the current color palette, or as one of the following **XlColorIndex** constants: **xlColorIndexAutomatic** or **xlColorIndexNone**. Read/write **Long**.

Remarks

Set this property to **xlColorIndexAutomatic** to specify the automatic pattern for cells or the automatic fill style for drawing objects. Set this property to **xlColorIndexNone** to specify that you don't want a pattern (this is the same as setting the **Pattern** property of the **Interior** object to **xlPatternNone**).

Remarks

The following illustration shows the color-index values in the default color palette.



PatternColorIndex Property Example

This example sets the color of the interior pattern for rectangle one on Sheet1.

```
With Worksheets("Sheet1").Rectangles(1).Interior
    .Pattern = xlChecker
    .PatternColorIndex = 5
End With
```

Received Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReceivedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReceivedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReceivedA "}

True if the workbook mailer has been received (if it's been sent by another user to the current user) and the current user hasn't modified the mailer by using the **Reply**, **ReplyAll**, or **ForwardMailer** method. PowerTalk requires that mailers be received before they can be forwarded or replied to. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read-only **Boolean**.

Received Property Example

This example displays the current status of the **Received** property.

```
With ActiveWorkbook
   If .HasMailer Then
      If .Mailer.Received Then
         state = "True"
      Else
         state = "False"
      End If
      MsgBox "Received property is " & state
   Else
      MsgBox "The workbook has no mailer"
   End If
End With
```

RegisterXLL Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRegisterXLLC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRegisterXLLX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRegisterXLLA "}

Loads an XLL code resource and automatically registers the functions and commands contained in the resource.

Syntax

expression.**RegisterXLL(*Filename*)**

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

Filename Required **String**. Specifies the name of the XLL to be loaded.

Remarks

This method returns **True** if the code resource is successfully loaded; otherwise, the method returns **False**.

RegisterXLL Method Example

This example loads an XLL file and registers the functions and commands in the file.

```
Application.RegisterXLL "XLMAPI.XLL"
```

Reply Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthReplyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthReplyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthReplyA "}

Replies to the workbook by creating a copy of the workbook and pre-initializing the new workbook's mailer to send to the originator of the workbook. Valid only when the workbook has a received mailer attached (you can only reply to a workbook you've received). Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed.

Syntax

expression.**Reply**

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

Remarks

To reply to a workbook, use this method to set up the mailer, use the **Mailer** property to adjust the mailer settings (if necessary), and then use the **SendMailer** method to send the reply.

This method generates an error if it's used in Windows.

Reply Method Example

This example replies to the sender of the active workbook.

```
Set original = ActiveWorkbook
If original.HasMailer Then
   original.Reply
   original.Mailer.Subject = "Here's my reply"
   ActiveWorkbook.SendMailer
End If
```

ReplyAll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthReplyAllC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthReplyAllX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthReplyAllA "}

Replies to the workbook by creating a copy of the workbook and pre-initializing the new workbook's mailer to send to all recipients of the workbook. Valid only when the workbook has a received mailer attached (you can only reply to a workbook you've received). Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed.

Syntax

expression.**ReplyAll**

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

Remarks

To reply to all recipients of a workbook, use this method to set up the mailer, use the **Mailer** property to adjust the mailer settings (if necessary), and then use the **SendMailer** method to send the reply.

This method generates an error if it's used in Windows.

ReplyAll Method Example

This example replies to all recipients of the active workbook.

```
Set original = ActiveWorkbook
If original.HasMailer Then
   original.ReplyAll
   original.Mailer.Subject = "Here's my reply"
   ActiveWorkbook.SendMailer
End If
```

Route Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRouteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRouteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRouteA "}

Routes the workbook, using the workbook's current routing slip.

Syntax

expression.**Route**

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

Remarks

Routing a workbook sets the **Routed** property to **True**.

Route Method Example

This example creates a routing slip for Book1.xls and then sends the workbook to three recipients, one after another.

```
Workbooks("BOOK1.XLS").HasRoutingSlip = True
With Workbooks("BOOK1.XLS").RoutingSlip
    .Delivery = xlOneAfterAnother
    .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
    .Subject = "Here is BOOK1.XLS"
    .Message = "Here is the workbook. What do you think?"
End With
Workbooks("BOOK1.XLS").Route
```

Routed Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRoutedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRoutedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRoutedA "}

True if the workbook has been routed to the next recipient. **False** if the workbook needs to be routed. Read-only **Boolean**.

Remarks

If the workbook wasn't routed to the current recipient, this property is always **False** (for example, if the document has no routing slip, or if a routing slip was just created).

Routed Property Example

This example sends the workbook to the next recipient.

```
If ActiveWorkbook.HasRoutingSlip And _
   Not ActiveWorkbook.Routed Then
      ActiveWorkbook.Route
End If
```

RoutingSlip Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRoutingSlipC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRoutingSlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRoutingSlipA "}

Returns a **RoutingSlip** object that represents the routing slip for the workbook. Reading this property if there's no routing slip causes an error (check the **HasRoutingSlip** property first). Read-only.

RoutingSlip Property Example

This example creates a routing slip for Book1.xls and then sends the workbook to three recipients, one after another.

```
Workbooks("BOOK1.XLS").HasRoutingSlip = True
With Workbooks("BOOK1.XLS").RoutingSlip
    .Delivery = xlOneAfterAnother
    .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
    .Subject = "Here is BOOK1.XLS"
    .Message = "Here is the workbook. What do you think?"
End With
Workbooks("BOOK1.XLS").Route
```

SaveCopyAs Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSaveCopyAsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSaveCopyAsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSaveCopyAsA "}

Saves a copy of the workbook to a file but doesn't modify the open workbook in memory.

Syntax

expression.**SaveCopyAs(*Filename*)**

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

Filename Required. Specifies the file name for the copy.

SaveCopyAs Method Example

This example saves a copy of the active workbook.

```
ActiveWorkbook.SaveCopyAs "C:\TEMP\XXXX.XLS"
```

This is the same example in Microsoft Excel for the Macintosh.

```
ActiveWorkbook.SaveCopyAs "HD:Temporary Folder:Temporary Workbook File"
```

SendDateTime Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSendDateTimeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSendDateTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSendDateTimeA "}

Returns the date and time that the mailer was sent. For this property to be valid, the mailer must have already been sent. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read-only **Date**.

SendDateTime Property Example

This example displays the sender of the workbook, plus the date and time it was sent.

```
If ActiveWorkbook.HasMailer Then
   MsgBox "This workbook was sent by " & _
      ActiveWorkbook.Mailer.Sender & " at " & _
      Format(ActiveWorkbook.Mailer.SendDateTime, "General Date")
End If
```

Sender Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSenderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSenderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSenderA "}

Returns the name of the user who sent this workbook mailer. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read-only **String**.

Sender Property Example

This example displays the sender of the workbook, plus the date and time it was sent.

```
If ActiveWorkbook.HasMailer Then
   MsgBox "This workbook was sent by " & _
      ActiveWorkbook.Mailer.Sender & " at " & _
      Format(ActiveWorkbook.Mailer.SendDateTime, "General Date")
End If
```

SendMail Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSendMailC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSendMailX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSendMailA "}

Sends the workbook by using the installed mail system.

Syntax

expression.**SendMail(*Recipients, Subject, ReturnReceipt*)**

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

Recipients Required **Variant**. Specifies the name of the recipient as text, or as an array of text strings if there are multiple recipients. At least one recipient must be specified, and all recipients are added as To recipients.

Subject Optional **Variant**. Specifies the subject of the message. If this argument is omitted, the document name is used.

ReturnReceipt Optional **Variant**. **True** to request a return receipt. **False** to not request a return receipt. The default value is **False**.

Remarks

Use the **SendMail** method in Microsoft Mail (MAPI or Microsoft Mail for the Macintosh) e-mail systems. Pass addressing information as parameters.

Use the **SendMailer** method in PowerTalk e-mail systems on the Macintosh. The **Mailer** object contains the addressing information for PowerTalk.

SendMail Method Example

This example sends the active workbook to a single recipient.

```
ActiveWorkbook.SendMail recipients:="Jean Selva"
```

SendMailer Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSendMailerC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSendMailerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSendMailerA "}

Sends the workbook by using the PowerTalk mailer. This method is available only on the Macintosh, with the PowerTalk system extension installed, and it can only be used on a workbook that has a mailer attached.

Syntax

expression.**SendMailer**(***FileFormat, Priority***)

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

FileFormat Optional **Variant**. Specifies the file format to use for the workbook that's sent. See the **FileFormat** property for a list of valid types.

Priority Optional **Variant**. Specifies the delivery priority of the message. Can be one of the following **XlPriority** constants: **xlPriorityNormal**, **xlPriorityHigh**, or **xlPriorityLow**. The default value is **xlPriorityNormal**.

Remarks

Use the **SendMail** method in Microsoft Mail (MAPI or Microsoft Mail for the Macintosh) e-mail systems. Pass addressing information as parameters.

Use the **SendMailer** method in PowerTalk e-mail systems on the Macintosh. The **Mailer** object contains the addressing information for PowerTalk.

SendMailer Method Example

This example sets up the **Mailer** object for workbook one and then sends the workbook.

```
With Workbooks(1)
    .HasMailer = True
    With .Mailer
        .Subject = "Here is the workbook"
        .ToRecipients = Array("Jean")
        .CCRecipients = Array("Adam", "Bernard")
        .BCCRecipients = Array("Chris")
        .Enclosures = Array("TestFile")
    End With
    .SendMailer
End With
```

ToRecipients Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproToRecipientsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproToRecipientsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproToRecipientsA "}

Returns or sets the direct recipients of the mailer. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read/write **Variant**.

Remarks

This property is an array of strings specifying the address, in one of the following formats:

- A record in the Preferred Personal Catalog. These names are one level deep ("Fred" or "June," for example).
- A full path specifying either a record in a personal catalog ("HD:Excel Folder:My Catalog:Barney") or a plain record ("HD:Folder:Martin").
- A relative path from the current working folder specifying either a personal catalog record ("My Catalog:Barney") or a plain record ("Martin").
- A path in a PowerShare catalog tree, in the form "CATALOG_NAME:<node>:RECORD_NAME", where <node> is a path to a PowerShare catalog. An example of a complete path is "AppleTalk:North Building Zone:George's Mac".

ToRecipients Property Example

This example sets up the **Mailer** object for workbook one and then sends the workbook.

```
With Workbooks(1)
    .HasMailer = True
    With .Mailer
        .Subject = "Here is the workbook"
        .ToRecipients = Array("Jean")
        .CCRecipients = Array("Adam", "Bernard")
        .BCCRecipients = Array("Chris")
        .Enclosures = Array("TestFile")
    End With
    .SendMailer
End With
```

UpdateFromFile Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUpdateFromFileC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUpdateFromFileX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUpdateFromFileA "}

Updates a read-only workbook from the saved disk version of the workbook if the disk version is more recent than the copy of the workbook that is loaded in memory. If the disk copy hasn't changed since the workbook was loaded, the in-memory copy of the workbook isn't reloaded.

Syntax

expression.**UpdateFromFile**

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

Remarks

This method is useful when a workbook is opened as read-only by user A and opened as read/write by user B. If user B saves a newer version of the workbook to disk while user A still has the workbook open, user A cannot get the updated copy without closing and reopening the workbook and losing view settings. The **UpdateFromFile** method updates the in-memory copy of the workbook from the disk file.

UpdateFromFile Method Example

This example updates the active workbook from the disk version of the file.

```
ActiveWorkbook.UpdateFromFile
```

Volatile Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthVolatileC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthVolatileX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthVolatileA "}

Marks a user-defined function as volatile. A volatile function must be recalculated whenever calculation occurs in any cells on the worksheet. A nonvolatile function is recalculated only when the input variables change. This method has no effect if it's not inside a user-defined function used to calculate a worksheet cell.

Syntax

expression.**Volatile(*Volatile*)**

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

Volatile Optional **Variant**. **True** to mark the function as volatile. **False** to mark the function as nonvolatile. The default value is **True**

Volatile Method Example

This example marks the user-defined function "My_Func" as volatile. The function will be recalculated whenever calculation occurs in any cells on the worksheet on which this function appears.

```
Function My_Func()
   Application.Volatile
   '
   '  Remainder of the function
   '
End Function
```

WindowNumber Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWindowNumberC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWindowNumberX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWindowNumberA "}

Returns the window number. For example, a window named "Book1.xls:2" has 2 as its window number. Most windows have the window number 1. Read-only **Long**.

Remarks

The window number isn't the same as the window index (the return value of the **Index** property), which is the position of the window within the **Windows** collection.

WindowNumber Property Example

This example creates a new window of the active window and then displays the window number of the new window.

```
ActiveWindow.NewWindow
MsgBox ActiveWindow.WindowNumber
```

XValues Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproXValuesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproXValuesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproXValuesA "}

Returns or sets an array of x values for a chart series. The **XValues** property can be set to a range on a worksheet or to an array of values, but it cannot be a combination of both. Read/write **Variant**.

XValues Property Example

This example sets the x values for series one in Chart1 to the range B1:B5 on Sheet1.

```
Charts("Chart1").SeriesCollection(1).XValues = _
      Worksheets("Sheet1").Range("B1:B5")
```

This example uses an array to set values for the individual points in series one in Chart1.

```
Charts("Chart1").SeriesCollection(1).XValues = _
      Array(5.0, 6.3, 12.6, 28, 50)
```

Formatting Codes for Headers and Footers

The following special formatting codes can be included as a part of the header and footer properties (**LeftHeader, CenterHeader, RightHeader, LeftFooter, CenterFooter, RightFooter**).

Format code	Description
&L	Left aligns the characters that follow.
&C	Centers the characters that follow.
&R	Right aligns the characters that follow.
&E	Turns double-underline printing on or off.
&X	Turns superscript printing on or off.
&Y	Turns subscript printing on or off.
&B	Turns bold printing on or off.
&I	Turns italic printing on or off.
&U	Turns underline printing on or off.
&S	Turns strikethrough printing on or off.
&O	Turns outline printing on or off (Macintosh only).
&H	Turns shadow printing on or off (Macintosh only).
&D	Prints the current date.
&T	Prints the current time.
&F	Prints the name of the document.
&A	Prints the name of the workbook tab.
&P	Prints the page number.
&P+number	Prints the page number plus the specified number.
&P-number	Prints the page number minus the specified number.
&&	Prints a single ampersand.
& "fontname"	Prints the characters that follow in the specified font. Be sure to include the double quotation marks.
&nn	Prints the characters that follow in the specified font size. Use a two-digit number to specify a size in points.
&N	Prints the total number of pages in the document.

AddCustomList Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddCustomListC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddCustomListX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddCustomListA "}

Adds a custom list for custom autofill and/or custom sort.

Syntax

expression.**AddCustomList(*ListArray, ByRow*)**

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

ListArray Required **Variant**. Specifies the source data, as either an array of strings or a **Range** object.

ByRow Optional **Variant**. Only used if ***ListArray*** is a **Range** object. **True** to create a custom list from each row in the range. **False** to create a custom list from each column in the range. If this argument is omitted and there are more rows than columns (or an equal number of rows and columns) in the range, Microsoft Excel creates a custom list from each column in the range. If this argument is omitted and there are more columns than rows in the range, Microsoft Excel creates a custom list from each row in the range.

Remarks

If the list you're trying to add already exists, this method does nothing.

AddCustomList Method Example

This example adds an array of strings as a custom list.

```
Application.AddCustomList Array("cogs", "sprockets", _
   "widgets", "gizmos")
```

AlertBeforeOverwriting Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAlertBeforeOverwritingC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAlertBeforeOverwritingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAlertBeforeOverwritingA "}

True if Microsoft Excel displays a message before overwriting nonblank cells during a drag-and-drop editing operation. Read/write **Boolean**.

AlertBeforeOverwriting Property Example

This example causes Microsoft Excel to display an alert before overwriting nonblank cells during drag-and-drop editing.

```
Application.AlertBeforeOverwriting = True
```

AutoFilter Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoFilterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoFilterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoFilterA "}

Syntax 1: Displays or hides the AutoFilter drop-down arrows.

Syntax 2: Filters a list using the AutoFilter.

Syntax 1

expression.**AutoFilter**

Syntax 2

expression.**AutoFilter(*Field, Criteria1, Operator, Criteria2*)**

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

Field Optional **Variant**. The integer offset of the field on which you want to base the filter (from the left of the list; the leftmost field is field one).

Criteria1 Optional **Variant**. The criteria (a string; for example, "101"). Use "=" to find blank fields, or use "<>" to find nonblank fields. If this argument is omitted, the criteria is All. If ***Operator*** is **xlTop10Items**, ***Criteria1*** specifies the number of items (for example, "10").

Operator Optional **Variant**. Can be one of the following **XlAutoFilterOperator** constants: **xlAnd**, **xlBottom10Items**, **xlBottom10Percent**, **xlOr**, **xlTop10Items**, or **xlTop10Percent**. Use **xlAnd** and **xlOr** with ***Criteria1*** and ***Criteria2*** to construct compound criteria.

Criteria2 Optional **Variant**. The second criteria (a string). Used with ***Criteria1*** and ***Operator*** to construct compound criteria.

AutoFilter Method Example

This example filters a list starting in cell A1 on Sheet1 to displayonly the entries in which field one is equal to the string "Otis".

```
Worksheets("Sheet1").Range("A1").AutoFilter _
   field:=1, _
   criteria1:="Otis"
```

AutomaticStyles Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutomaticStylesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutomaticStylesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutomaticStylesA "}

True if the outline uses automatic styles. Read/write **Boolean**.

AutomaticStyles Property Example

This example sets the outline on Sheet1 to use automatic styles.

```
Worksheets("Sheet1").Outline.AutomaticStyles = True
```

BottomMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBottomMarginC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBottomMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBottomMarginA "}

Returns or sets the size of the bottom margin, in points. Read/write **Double**.

Remarks

Margins are set or returned in points. Use either the **InchesToPoints** method or the **CentimetersToPoints** method to do the conversion.

BottomMargin Property Example

These two examples set the bottom margin of Sheet1 to 0.5 inch (36 points).

```
Worksheets("Sheet1").PageSetup.BottomMargin = _
      Application.InchesToPoints(0.5)

Worksheets("Sheet1").PageSetup.BottomMargin = 36
```

This example displays the current setting for the bottom margin on Sheet1.

```
marginInches = Worksheets("Sheet1").PageSetup.BottomMargin / _
   Application.InchesToPoints(1)
MsgBox "The current bottom margin is " & marginInches & " inches"
```

CenterFooter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCenterFooterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCenterFooterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCenterFooterA "}

Returns or sets the center part of the footer. Read/write **String**.

Remarks

Special format codes can be used in the footer text.

CenterFooter Property Example

This example prints the workbook name and page number at the bottom of each page.

```
Worksheets("Sheet1").PageSetup.CenterFooter = "&F page &P"
```

CenterHeader Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCenterHeaderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCenterHeaderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCenterHeaderA "}

Returns or sets the center part of the header. Read/write **String**.

Remarks

Special format codes can be used in the header text.

CenterHeader Property Example

This example prints the date and page number at the top of each page.

```
Worksheets("Sheet1").PageSetup.CenterHeader = "&D page &P of &N"
```

CenterHorizontally Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCenterHorizontallyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCenterHorizontallyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCenterHorizontallyA "}

True if the sheet is centered horizontally on the page when it's printed. Read/write **Boolean**.

CenterHorizontally Property Example

This example centers Sheet1 horizontally when it'sprinted.

```
Worksheets("Sheet1").PageSetup.CenterHorizontally = True
```

CenterVertically Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCenterVerticallyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCenterVerticallyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCenterVerticallyA "}

True if the sheet is centered vertically on the page when it's printed. Read/write **Boolean**.

CenterVertically Property Example

This example centers Sheet1 vertically when it's printed.

```
Worksheets("Sheet1").PageSetup.CenterVertically = True
```

ChartSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChartSizeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChartSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChartSizeA "}

Returns or sets the way a chart is scaled to fit on a page. Read/write **Long**.

Can be one of the following **XlObjectSize** constants.

Constant	Meaning
xlScreenSize	Print the chart the same size as it appears on the screen.
xlFitToPage	Print the chart as large as possible, while retaining the chart's height-to-width ratio as shown on the screen.
xlFullPage	Print the chart to fit the page, adjusting the height-to-width ratio as necessary.

Remarks

This property applies only to chart sheets (it cannot be used with embedded charts).

ChartSize Property Example

This example scales the first chart in the active workbook to fit a full page.

```
ActiveWorkbook.Charts(1).PageSetup.ChartSize = xlFullPage
```

DeleteCustomList Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDeleteCustomListC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDeleteCustomListX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDeleteCustomListA "}

Deletes a custom list.

Syntax

expression.**DeleteCustomList(*ListNum*)**

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

ListNum Required **Long**. The custom list number. This number must be greater than or equal to 5 (Microsoft Excel has four built-in custom lists that cannot be deleted).

Remarks

This method generates an error if the list number is less than 5 or if there's no matching custom list.

DeleteCustomList Method Example

This example deletes a custom list.

```
n = Application.GetCustomListNum(Array("cogs", "sprockets", _
   "widgets", "gizmos"))
Application.DeleteCustomList n
```

DisplayHorizontalScrollBar Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayHorizontalScrollBarC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayHorizontalScrollBarX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayHorizontalScrollBarA "}

True if the horizontal scroll bar is displayed. Read/write **Boolean**.

DisplayHorizontalScrollBar Property Example

This example turns on the horizontal scroll bar for the active window.

```
ActiveWindow.DisplayHorizontalScrollBar = True
```

DisplayVerticalScrollBar Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayVerticalScrollBarC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayVerticalScrollBarX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayVerticalScrollBarA "}

True if the vertical scroll bar is displayed. Read/write **Boolean**.

DisplayVerticalScrollBar Property Example

This example turns on the vertical scroll bar for the active window.

```
ActiveWindow.DisplayVerticalScrollBar = True
```

DisplayWorkbookTabs Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayWorkbookTabsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayWorkbookTabsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayWorkbookTabsA "}

True if the workbook tabs are displayed. Read/write **Boolean**.

DisplayWorkbookTabs Property Example

This example turns on the workbook tabs.

```
ActiveWindow.DisplayWorkbookTabs = True
```

Draft Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDraftC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDraftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDraftA "}

True if the sheet will be printed without graphics. Read/write **Boolean**.

Remarks

Setting this property to **True** makes printing faster (at the expense of not printing graphics).

Draft Property Example

This example turns off graphics printing for Sheet1.

```
Worksheets("Sheet1").PageSetup.Draft = True
```

EditDirectlyInCell Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEditDirectlyInCellC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEditDirectlyInCellX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEditDirectlyInCellA "}

True if Microsoft Excel allows editing in cells. Read/write **Boolean**.

EditDirectlyInCell Property Example

This example enables editing in cells.

```
Application.EditDirectlyInCell = True
```

FilterMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFilterModeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFilterModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFilterModeA "}

True if the worksheet is in filter mode. Read-only **Boolean**.

Remarks

This property is **True** if the worksheet contains a filtered list in which there are hidden rows.

FilterMode Property Example

This example displays the filter status of Sheet1 in a message box.

```
If Worksheets("Sheet1").FilterMode = True Then
   MsgBox "Filter mode is on"
Else
   MsgBox "Filter mode is off"
End If
```

FirstPageNumber Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFirstPageNumberC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFirstPageNumberX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFirstPageNumberA "}

Returns or sets the first page number that will be used when this sheet is printed. If **xlAutomatic**, Microsoft Excel chooses the first page number. The default is **xlAutomatic**. Read/write **Long**.

FirstPageNumber Property Example

This example sets the first page number of Sheet1 to 100.

```
Worksheets("Sheet1").PageSetup.FirstPageNumber = 100
```

FitToPagesTall Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFitToPagesTallC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFitToPagesTallX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFitToPagesTallA "}

Returns or sets the number of pages tall the worksheet will be scaled to when it's printed. Applies only to worksheets. Read/write **Variant**.

Remarks

If this property is **False**, Microsoft Excel scales the worksheet according to the **FitToPagesWide** property.

If the **Zoom** property is **True**, the **FitToPagesTall** property is ignored.

FitToPagesTall Property Example

This example causes Microsoft Excel to print Sheet1 exactly one page tall and wide.

```
With Worksheets("Sheet1").PageSetup
    .Zoom = False
    .FitToPagesTall = 1
    .FitToPagesWide = 1
End With
```

FitToPagesWide Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFitToPagesWideC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFitToPagesWideX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFitToPagesWideA "}

Returns or sets the number of pages wide the worksheet will be scaled to when it's printed. Applies only to worksheets. Read/write **Variant**.

Remarks

If this property is **False**, Microsoft Excel scales the worksheet according to the **FitToPagesTall** property.

If the **Zoom** property is **True**, the **FitToPagesWide** property is ignored.

FitToPagesWide Property Example

This example causes Microsoft Excel to print Sheet1 exactly one page wide and tall.

```
With Worksheets("Sheet1").PageSetup
    .Zoom = False
    .FitToPagesTall = 1
    .FitToPagesWide = 1
End With
```

FontStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFontStyleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFontStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFontStyleA "}

Returns or sets the font style. Read/write **String**.

Remarks

Changing this property may affect other **Font** properties (such as **Bold** and **Italic**).

FontStyle Property Example

This example sets the font style for cell A1 on Sheet1 to bold and italic.

```
Worksheets("Sheet1").Range("A1").Font.FontStyle = "Bold Italic"
```

FooterMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFooterMarginC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFooterMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFooterMarginA "}

Returns or sets the distance from the bottom of the page to the footer, in points. Read/write **Double**.

FooterMargin Property Example

This example sets the footer margin of Sheet1 to 0.5 inch.

```
Worksheets("Sheet1").PageSetup.FooterMargin = _
      Application.InchesToPoints(0.5)
```

GetCustomListContents Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGetCustomListContentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGetCustomListContentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGetCustomListContentsA "}

Returns a custom list (an array of strings).

Syntax

expression.**GetCustomListContents(*ListNum*)**

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

ListNum Required **Long**. The list number.

GetCustomListContents Method Example

This example writes the elements of the first custom list in column one on Sheet1.

```
listArray = Application.GetCustomListContents(1)
For i = LBound(listArray, 1) To UBound(listArray, 1)
   Worksheets("sheet1").Cells(i, 1).Value = listArray(i)
Next i
```

GetCustomListNum Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGetCustomListNumC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGetCustomListNumX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGetCustomListNumA "}

Returns the custom list number for an array of strings. You can use this method to match both built-in lists and custom-defined lists.

Syntax

expression.**GetCustomListNum(*ListArray*)**

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

ListArray Required **Variant**. An array of strings.

Remarks

This method generates an error if there's no corresponding list.

GetCustomListNum Method Example

This example deletes a custom list.

```
n = Application.GetCustomListNum(Array("cogs", "sprockets", _
   "widgets", "gizmos"))
Application.DeleteCustomList n
```

HeaderMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHeaderMarginC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHeaderMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHeaderMarginA "}

Returns or sets the distance from the top of the page to the header, in points. Read/write **Double**.

Remarks

Margins are set or returned in points. Use the **InchesToPoints** method or the **CentimetersToPoints** method to convert measurements from inches or centimeters.

HeaderMargin Property Example

This example sets the header margin of Sheet1 to 0.5 inch.

```
Worksheets("Sheet1").PageSetup.HeaderMargin = _
      Application.InchesToPoints(0.5)
```

LeftFooter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLeftFooterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLeftFooterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLeftFooterA "}

Returns or sets the left part of the footer. Read/write **String**.

Remarks

Special format codes can be used in the footer text.

LeftFooter Property Example

This example prints the page number in the lower-left corner of every page.

```
Worksheets("Sheet1").PageSetup.LeftFooter = "&P"
```

LeftHeader Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLeftHeaderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLeftHeaderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLeftHeaderA "}

Returns or sets the left part of the header. Read/write **String**.

Remarks

Special format codes can be used in the header text.

LeftHeader Property Example

This example prints the date in the upper-left corner of every page.

```
Worksheets("Sheet1").PageSetup.LeftHeader = "&D"
```

LeftMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLeftMarginC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLeftMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLeftMarginA "}

Returns or sets the size of the left margin, in points. Read/write **Double**.

Remarks

Margins are set or returned in points. Use the **InchesToPoints** method or the **CentimetersToPoints** method to convert measurements from inches or centimeters.

LeftMargin Property Example

This example sets the left margin of Sheet1 to 1.5 inches.

```
Worksheets("Sheet1").PageSetup.LeftMargin = _
      Application.InchesToPoints(1.5)
```

This example sets the left margin of Sheet1 to 2 centimeters.

```
Worksheets("Sheet1").PageSetup.LeftMargin = _
      Application.CentimetersToPoints(2)
```

This example displays the current left-margin setting for Sheet1.

```
marginInches = Worksheets("Sheet1").PageSetup.LeftMargin / _
   Application.InchesToPoints(1)
MsgBox "The current left margin is " & marginInches & " inches"
```

MailLogoff Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthMailLogoffC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthMailLogoffX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthMailLogoffA "}

Closes a MAPI mail session established by Microsoft Excel.

Syntax

expression.**MailLogoff**

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

Remarks

You cannot use this method to close or log off Microsoft Mail.

MailLogoff Method Example

This example closes the established mail session, if there is one.

```
If Not IsNull(Application.MailSession) Then Application.MailLogoff
```

MailLogon Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthMailLogonC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthMailLogonX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthMailLogonA "}

Logs in to MAPI Mail or Microsoft Exchange and establishes a mail session. If Microsoft Mail isn't already running, you must use this method to establish a mail session before mail or document routing functions can be used.

Syntax

expression.**MailLogon(*Name, Password, DownloadNewMail*)**

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

Name Optional **Variant**. The mail account name or Microsoft Exchange profile name. If this argument is omitted, the default mail account name is used.

Password Optional **Variant**. The mail account password. This argument is ignored in Microsoft Exchange.

DownloadNewMail Optional **Variant**. **True** to download new mail immediately.

Remarks

Microsoft Excel logs off any mail sessions it previously established before attempting to establish the new session.

To piggyback on the system default mail session, omit both the name and password parameters.

MailLogon Method Example

This example logs in to mail and downloads any new mail immediately.

```
If IsNull(Application.MailSession) Then
   Application.MailLogon "oscarx", "mypassword", True
End If
```

MailSession Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMailSessionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMailSessionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMailSessionA "}

Returns the MAPI mail session number as a hexadecimal string (if there's an active session), or returns **Null** if there's no session. Read-only **Variant**.

Remarks

This property applies only to mail sessions created by Microsoft Excel (it doesn't return a mail session number for Microsoft Mail).

This property isn't used on PowerTalk mail systems.

MailSession Property Example

This example closes the established mail session, if there is one.

```
If Not IsNull(Application.MailSession) Then Application.MailLogoff
```

MailSystem Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMailSystemC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMailSystemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMailSystemA "}

Returns the mail system that's installed on the host machine. Can be one of the following **XlMailSystem** constants: **xlNoMailSystem**, **xlMAPI**, or **xlPowerTalk**. Read-only **Long**.

MailSystem Property Example

This example displays the name of the mail system that's installed on the computer.

```
Select Case Application.MailSystem
   Case xlMAPI
      MsgBox "Mail system is Microsoft Mail"
   Case xlPowerTalk
      MsgBox "Mail system is PowerTalk"
   Case xlNoMailSystem
      MsgBox "No mail system installed"
End Select
```

NextLetter Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthNextLetterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthNextLetterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthNextLetterA "}

Opens the oldest unread Microsoft Excel letter from the In Tray. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed.

Syntax

expression.**NextLetter**

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

Remarks

This method returns a **Workbook** object for the newly opened workbook, or it returns **Null** if there are no more workbooks to open.

This method generates an error if it's used in Windows.

NextLetter Method Example

This example opens the oldest unread Microsoft Excel letter from the In Tray.

```
If Application.MailSystem = xlPowerTalk Then _
   Application.NextLetter
```

Order Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOrderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOrderA "}

PageSetup object: Returns or sets the order that Microsoft Excel uses to number pages when printing a large worksheet. Can be one of the following **XlOrder** constants: **xlDownThenOver** or **xlOverThenDown**. Applies only to worksheets. Read/write **Long**.

Trendline object: Returns or sets the trendline order (an integer greater than 1) when the trendline type is **xlPolynomial**. Read/write **Long**.

Order Property Example

This example breaks Sheet1 into pages when the worksheet is printed. Numbering and printing proceed from the first page to the pages to the right, and then move down and continue printing across the sheet.

```
Worksheets("Sheet1").PageSetup.Order = xlOverThenDown
```

PaperSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPaperSizeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPaperSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPaperSizeA "}

Windows only. Returns or sets the size of the paper. Read/write **Long**.

Can be one of the following **XlPaperSize** constants. (Some printers may not support all of these paper sizes.)

Constant	Meaning
xlPaperLetter	Letter (8-1/2 in. x 11 in.)
xlPaperLetterSmall	Letter Small (8-1/2 in. x 11 in.)
xlPaperTabloid	Tabloid (11 in. x 17 in.)
xlPaperLedger	Ledger (17 in. x 11 in.)
xlPaperLegal	Legal (8-1/2 in. x 14 in.)
xlPaperStatement	Statement (5-1/2 in. x 8-1/2 in.)
xlPaperExecutive	Executive (7-1/2 in. x 10-1/2 in.)
xlPaperA3	A3 (297 mm x 420 mm)
xlPaperA4	A4 (210 mm x 297 mm)
xlPaperA4Small	A4 Small (210 mm x 297 mm)
xlPaperA5	A5 (148 mm x 210 mm)
xlPaperB4	B4 (250 mm x 354 mm)
xlPaperB5	B5 (182 mm x 257 mm)
xlPaperFolio	Folio (8-1/2 in. x 13 in.)
xlPaperQuarto	Quarto (215 mm x 275 mm)
xlPaper10x14	10 in. x 14 in.
xlPaper11x17	11 in. x 17 in.
xlPaperNote	Note (8-1/2 in. x 11 in.)
xlPaperEnvelope9	Envelope #9 (3-7/8 in. x 8-7/8 in.)
xlPaperEnvelope10	Envelope #10 (4-1/8 in. x 9-1/2 in.)
xlPaperEnvelope11	Envelope #11 (4-1/2 in. x 10-3/8 in.)
xlPaperEnvelope12	Envelope #12 (4-1/2 in. x 11 in.)
xlPaperEnvelope14	Envelope #14 (5 in. x 11-1/2 in.)
xlPaperCsheet	C size sheet
xlPaperDsheet	D size sheet
xlPaperEsheet	E size sheet
xlPaperEnvelopeDL	Envelope DL (110 mm x 220 mm)
xlPaperEnvelopeC3	Envelope C3 (324 mm x 458 mm)
xlPaperEnvelopeC4	Envelope C4 (229 mm x 324 mm)
xlPaperEnvelopeC5	Envelope C5 (162 mm x 229 mm)
xlPaperEnvelopeC6	Envelope C6 (114 mm x 162 mm)
xlPaperEnvelopeC65	Envelope C65 (114 mm x 229 mm)
xlPaperEnvelopeB4	Envelope B4 (250 mm x 353 mm)
xlPaperEnvelopeB5	Envelope B5 (176 mm x 250 mm)
xlPaperEnvelopeB6	Envelope B6 (176 mm x 125 mm)
xlPaperEnvelopeItaly	Envelope (110 mm x 230 mm)
xlPaperEnvelopeMonarch	Envelope Monarch (3-7/8 in. x 7-1/2 in.)

xlPaperEnvelopePersonal	Envelope (3-5/8 in. x 6-1/2 in.)
xlPaperFanfoldUS	U.S. Standard Fanfold (14-7/8 in. x 11 in.)
xlPaperFanfoldStdGerman	German Standard Fanfold (8-1/2 in. x 12 in.)
xlPaperFanfoldLegalGerman	German Legal Fanfold (8-1/2 in. x 13 in.)
xlPaperUser	User-defined

PaperSize Property Example

This example sets the paper size to legal for Sheet1.

```
Worksheets("Sheet1").PageSetup.PaperSize = xlPaperLegal
```

PrintGridlines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintGridlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintGridlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintGridlinesA "}

True if cell gridlines are printed on the page. Applies only to worksheets. Read/write **Boolean**.

PrintGridlines Property Example

This example prints cell gridlines when Sheet1 is printed.

```
Worksheets("Sheet1").PageSetup.PrintGridlines = True
```

PrintHeadings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintHeadingsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintHeadingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintHeadingsA "}

True if row and column headings are printed with this page. Applies only to worksheets. Read/write **Boolean**.

Remarks

The **DisplayHeadings** property controls the on-screen display of headings.

PrintHeadings Property Example

This example turns off the printing of headings for Sheet1.

```
Worksheets("Sheet1").PageSetup.PrintHeadings = False
```

PrintNotes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintNotesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintNotesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintNotesA "}

True if cell notes are printed as end notes with the sheet. Applies only to worksheets. Read/write **Boolean**.

Remarks

Use the **PrintComments** property to print comments as text boxes or end notes.

PrintNotes Property Example

This example turns off the printing of notes.

```
Worksheets("Sheet1").PageSetup.PrintNotes = False
```

PromptForSummaryInfo Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPromptForSummaryInfoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPromptForSummaryInfoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPromptForSummaryInfoA "}

True if Microsoft Excel asks for summary information when files are first saved. Read/write **Boolean**.

PromptForSummaryInfo Property Example

This example displays a prompt that asks for summary information when files are first saved.

```
Application.PromptForSummaryInfo = True
```

RightFooter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRightFooterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRightFooterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRightFooterA "}

Returns or sets the right part of the footer. Read/write **String**.

Remarks

Special format codes can be used in the footer text.

RightFooter Property Example

This example prints the page number in the lower-right corner of every page.

```
Worksheets("Sheet1").PageSetup.RightFooter = "&P"
```

RightHeader Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRightHeaderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRightHeaderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRightHeaderA "}

Returns or sets the right part of the header. Read/write **String**.

Remarks

Special format codes can be used in the header text.

RightHeader Property Example

This example prints the filename in the upper-right corner of every page.

```
Worksheets("Sheet1").PageSetup.RightHeader = "&F"
```

RightMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRightMarginC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRightMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRightMarginA "}

Returns or sets the size of the right margin, in points. Read/write **Double**.

Remarks

Margins are set or returned in points. Use the **InchesToPoints** method or the **CentimetersToPoints** method to convert measurements from inches or centimeters.

RightMargin Property Example

This example sets the right margin of Sheet1 to 1.5 inches.

```
Worksheets("Sheet1").PageSetup.RightMargin = _
      Application.InchesToPoints(1.5)
```

This example sets the right margin of Sheet1 to 2 centimeters.

```
Worksheets("Sheet1").PageSetup.RightMargin = _
      Application.CentimetersToPoints(2)
```

This example displays the current right-margin setting for Sheet1.

```
marginInches = Worksheets("Sheet1").PageSetup.RightMargin / _
   Application.InchesToPoints(1)
MsgBox "The current right margin is " & marginInches & " inches"
```

ScrollColumn Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproScrollColumnC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproScrollColumnX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproScrollColumnA "}

Returns or sets the number of the leftmost column in the pane or window. Read/write **Long**.

Remarks

If the window is split, the **ScrollColumn** property of the **Window** object refers to the upper-left pane. If the panes are frozen, the **ScrollColumn** property of the **Window** object excludes the frozen areas.

ScrollColumn Property Example

This example moves column three so that it's the leftmost column in the window.

```
Worksheets("Sheet1").Activate
ActiveWindow.ScrollColumn = 3
```

ScrollRow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproScrollRowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproScrollRowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproScrollRowA "}

Returns or sets the number of the row that appears at the top of the pane or window. Read/write **Long**.

Remarks

If the window is split, the **ScrollRow** property of the **Window** object refers to the upper-left pane. If the panes are frozen, the **ScrollRow** property of the **Window** object excludes the frozen areas.

ScrollRow Property Example

This example moves row ten to the top of the window.

```
Worksheets("Sheet1").Activate
ActiveWindow.ScrollRow = 10
```

ScrollWorkbookTabs Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthScrollWorkbookTabsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthScrollWorkbookTabsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthScrollWorkbookTabsA "}

Scrolls through the workbook tabs at the bottom of the window. Doesn't affect the active sheet in the workbook.

Syntax

expression.**ScrollWorkbookTabs(*Sheets, Position*)**

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

Sheets Optional **Variant**. The number of sheets to scroll by. Use a positive number to scroll forward, a negative number to scroll backward, or 0 (zero) to not scroll at all. You must specify ***Sheets*** if you don't specify ***Position***.

Position Optional **Variant**. Use **xlFirst** to scroll to the first sheet, or use **xlLast** to scroll to the last sheet. You must specify ***Position*** if you don't specify ***Sheets***.

ScrollWorkbookTabs Method Example

This example scrolls through the workbook tabs to the last sheet in the workbook.

```
ActiveWindow.ScrollWorkbookTabs position:=xlLast
```

SheetsInNewWorkbook Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSheetsInNewWorkbookC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSheetsInNewWorkbookX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSheetsInNewWorkbookA "}

Returns or sets the number of sheets that Microsoft Excel automatically inserts into new workbooks. Read/write **Long**.

SheetsInNewWorkbook Property Example

This example displays the number of sheets automatically inserted into new workbooks.

```
MsgBox "Microsoft Excel inserts " & _
   Application.SheetsInNewWorkbook & _
   " sheet(s) in each new workbook"
```

ShowLevels Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowLevelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowLevelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowLevelsA "}

Displays the specified number of row and/or column levels of an outline.

Syntax

expression.**ShowLevels(*RowLevels, ColumnLevels*)**

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

RowLevels Optional **Variant**. Specifies the number of row levels of an outline to display. If the outline has fewer levels than the number specified, Microsoft Excel displays all the levels. If this argument is 0 (zero) or is omitted, no action is taken on rows.

ColumnLevels Optional **Variant**. Specifies the number of column levels of an outline to display. If the outline has fewer levels than the number specified, Microsoft Excel displays all the levels. If this argument is 0 (zero) or is omitted, no action is taken on columns.

Remarks

You must specify at least one argument.

ShowLevels Method Example

This example displays row levels one through three and column level one of the outline on Sheet1.

```
Worksheets("Sheet1").Outline.ShowLevels rowLevels:=3, columnLevels:=1
```

StandardFont Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStandardFontC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStandardFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStandardFontA "}

Returns or sets the name of the standard font. Read/write **String**.

Remarks

If you change the standard font by using this property, the change doesn't take effect until you restart Microsoft Excel.

StandardFont Property Example

This example sets the standard font to Geneva (on the Macintosh) or Arial (in Windows).

```
If Application.OperatingSystem Like "*Macintosh*" Then
   Application.StandardFont = "Geneva"
Else
   Application.StandardFont = "Arial"
End If
```

StandardFontSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStandardFontSizeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStandardFontSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStandardFontSizeA "}

Returns or sets the standard font size, in points. Read/write **Long**.

Remarks

If you change the standard font size by using this property, the change doesn't take effect until you restart Microsoft Excel.

StandardFontSize Property Example

This example sets the standard font size to 12 points.

```
Application.StandardFontSize = 12
```

Subscript Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSubscriptC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSubscriptX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSubscriptA "}

True if the font is formatted as subscript. **False** by default. Read/write **Variant**.

Subscript Property Example

This example makes the second character in cell A1 a subscript character.

```
Worksheets("Sheet1").Range("A1").Characters(2, 1).Font.Subscript = True
```

SummaryColumn Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSummaryColumnC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSummaryColumnX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSummaryColumnA "}

Returns or sets the location of the summary columns in the outline, as shown in the following table. Read/write **Long**.

Value	Meaning
xlLeft	The summary column will be positioned to the left of the detail columns in the outline.
xlRight	The summary column will be positioned to the right of the detail columns in the outline.

SummaryColumn Property Example

This example creates an outline with automatic styles, with the summary row above the detail rows, and with the summary column to the right of the detail columns.

```
Worksheets("Sheet1").Activate
Selection.AutoOutline
With ActiveSheet.Outline
    .SummaryRow = xlAbove
    .SummaryColumn = xlRight
    .AutomaticStyles = True
End With
```

SummaryRow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSummaryRowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSummaryRowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSummaryRowA "}

Returns or sets the location of the summary rows in the outline, as shown in the following table. Read/write **Long**.

Value	Meaning
xlAbove	The summary row will be positioned above the detail rows in the outline.
xlBelow	The summary row will be positioned below the detail rows in the outline.

Remarks

Set **SummaryRow** to **xlAbove** for Microsoft Word-style outlines, where category headers are above the detailed information. Set **SummaryRow** to **xlBelow** for accounting-style outlines, where summations are below the detailed information.

SummaryRow Property Example

This example creates an outline with automatic styles, with the summary row above the detail rows, and with the summary column to the right of the detail columns.

```
Worksheets("Sheet1").Activate
Selection.AutoOutline
With ActiveSheet.Outline
    .SummaryRow = xlAbove
    .SummaryColumn = xlRight
    .AutomaticStyles = True
End With
```

Superscript Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSuperscriptC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSuperscriptX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSuperscriptA "}

True if the font is formatted as superscript; **False** by default. Read/write **Variant**.

Superscript Property Example

This example makes the last character in cell A1 a superscript character.

```
n = Worksheets("Sheet1").Range("A1").Characters.Count
Worksheets("Sheet1").Range("A1").Characters(n, 1).Font.Superscript = _
         True
```

TabRatio Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTabRatioC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTabRatioX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTabRatioA "}

Returns or sets the ratio of the width of the workbook's tab area to the width of the window's horizontal scroll bar (as a number between 0 (zero) and 1; the default value is 0.75). Read/write **Double**.

Remarks

This property has no effect when **DisplayWorkbookTabs** is set to **False** (its value is retained, but it has no effect on the display).

TabRatio Property Example

This example makes the workbook tabs half the width of the horizontal scroll bar.

```
ActiveWindow.TabRatio = 0.5
```

TopMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTopMarginC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTopMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTopMarginA "}

Returns or sets the size of the top margin, in points. Read/write **Double**.

Remarks

Margins are set or returned in points. Use the **InchesToPoints** method or the **CentimetersToPoints** method to convert measurements from inches or centimeters.

TopMargin Property Example

These two examples set the top margin of Sheet1 to 0.5 inch (36 points).

```
Worksheets("Sheet1").PageSetup.TopMargin = _
      Application.InchesToPoints(0.5)

Worksheets("Sheet1").PageSetup.TopMargin = 36
```

This example displays the current top-margin setting.

```
marginInches = ActiveSheet.PageSetup.TopMargin / _
   Application.InchesToPoints(1)
MsgBox "The current top margin is " & marginInches & " inches"
```

Zoom Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproZoomC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproZoomX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproZoomA "}

PageSetup object:

Returns or sets a percentage (between 10 and 400 percent) by which Microsoft Excel will scale the worksheet for printing. Applies only to worksheets. Read/write **Variant**.

If this property is **False**, the **FitToPagesWide** and **FitToPagesTall** properties control how the worksheet is scaled.

Window object:

Returns or sets the display size of the window, as a percentage (100 equals normal size, 200 equals double size, and so on). Read/write **Variant**.

You can also set this property to **True** to make the window size fit the current selection.

Remarks

PageSetup object:

All scaling retains the aspect ratio of the original document.

Window object:

This function affects only the sheet that's currently active in the window. To use this property on other sheets, you must first activate them.

Zoom Property Example

This example scales Sheet1 by 150 percent when the worksheet is printed.

```
Worksheets("Sheet1").PageSetup.Zoom = 150
```

point

A unit of measurement equal to 1/72 inch.

ActiveChart Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproActiveChartC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproActiveChartX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproActiveChartA "}

Returns a **Chart** object that represents the active chart (either an embedded chart or a chart sheet). An embedded chart is considered active when it's either selected or activated. When no chart is active, this property returns **Nothing**. Read-only.

Remarks

If you don't specify an object qualifier, this property returns the active chart in the active workbook.

ActiveChart Property Example

This example turns on the legend for the active chart.

```
ActiveChart.HasLegend = True
```

AddChartAutoFormat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddChartAutoFormatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddChartAutoFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddChartAutoFormatA "}

Adds a custom chart autoformat to the list of available chart autoformats.

Syntax

expression.**AddChartAutoFormat(*Chart, Name, Description*)**

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

Chart Required **Chart**. A chart that contains the format that will be applied when the new chart autoformat is applied.

Name Required **String**. The name of the autoformat.

Description Optional **String**. A description of the custom autoformat.

AddChartAutoFormat Method Example

This example adds a new autoformat based on Chart1.

```
Application.AddChartAutoFormat _
   Chart:=Charts("Chart1"), Name:="Presentation Chart"
```

AdvancedFilter Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAdvancedFilterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAdvancedFilterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAdvancedFilterA "}

Filters or copies data from a list based on a criteria range. If the initial selection is a single cell, that cell's current region is used.

Syntax

expression.**AdvancedFilter(*Action, CriteriaRange, CopyToRange, Unique*)**

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

Action Required **Long**. The filter operation. Can be one of the following **XlFilterAction** constants: **xlFilterInPlace** or **xlFilterCopy**.

CriteriaRange Optional **Variant**. The criteria range. If this argument is omitted, there are no criteria.

CopyToRange Optional **Variant**. The destination range for the copied rows if ***Action*** is **xlFilterCopy**. Otherwise, this argument is ignored.

Unique Optional **Variant**. **True** to filter unique records only. **False** to filter all records that meet the criteria. The default value is **False**.

AdvancedFilter Method Example

This example filters a database (named "Database") based on a criteria range named "Criteria."

```
Range("Database").AdvancedFilter _
   Action:=xlFilterInPlace, _
   CriteriaRange:=Range("Criteria")
```

AutoFilterMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoFilterModeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoFilterModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoFilterModeA "}

True if the AutoFilter drop-down arrows are currently displayed on the sheet. This property is independent of the **FilterMode** property. Read/write **Boolean**.

Remarks

This property returns **True** if the drop-down arrows are currently displayed. You can set this property to **False** to remove the arrows, but you cannot set it to **True**. Use the **AutoFilter** method to filter a list and display the drop-down arrows.

AutoFilterMode Property Example

This example displays the current status of the **AutoFilterMode** property on Sheet1.

```
If Worksheets("Sheet1").AutoFilterMode Then
   isOn = "On"
Else
   isOn = "Off"
End If
MsgBox "AutoFilterMode is " & isOn
```

AutoOutline Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoOutlineC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoOutlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoOutlineA "}

Automatically creates an outline for the specified range. If the range is a single cell, Microsoft Excel creates an outline for the entire sheet. The new outline replaces any existing outline.

Syntax

expression.**AutoOutline**

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

AutoOutline Method Example

This example creates an outline for the range A1:G37 on Sheet1. The range must contain either a summary row or a summary column.

```
Worksheets("Sheet1").Range("A1:G37").AutoOutline
```

BorderAround Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthBorderAroundC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthBorderAroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthBorderAroundA "}

Adds a border to a range and sets the **Color**, **LineStyle**, and **Weight** properties for the new border.

Syntax

expression.**BorderAround(*LineStyle, Weight, ColorIndex, Color*)**

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

LineStyle Optional **Variant**. The line style for the border. Can be one of the following **XlBorderLineStyle** constants: **xlBorderLineStyleNone**, **xlBorderLineStyleContinuous**, **xlBorderLineStyleDash**, **xlBorderLineStyleDot**, or **xlBorderLineStyleDouble**. The default value is **xlBorderLineStyleContinuous**.

Weight Optional **Variant**. The border weight. Can be one of the following **XlBorderWeight** constants: **xlHairline**, **xlThin**, **xlMedium**, or **xlThick**. The default value is **xlThin**.

ColorIndex Optional **Variant**. The border color, as an index into the current color palette, or as one of the following **XlColorIndex** constants: **xlColorIndexAutomatic** or **xlColorIndexNone**.

Color Optional **Variant**. The border color, as an RGB value.

Remarks

You can specify either ***ColorIndex*** or ***Color***, but not both. If you don't specify either argument, Microsoft Excel uses the **xlAutomatic** color index.

Similarly, you can specify either ***LineStyle*** or ***Weight***, but not both. If you don't specify either argument, Microsoft Excel creates a default border.

This method outlines the entire range without filling it in. To set the borders of all the cells, you must set the **Color**, **LineStyle**, and **Weight** properties for the **Borders** collection. To clear the border, you must set the **LineStyle** property to **xlBorderLineStyleNone** for all the cells in the range.

BorderAround Method Example

This example adds a thick red border around the range A1:D4 on Sheet1.

```
Worksheets("Sheet1").Range("A1:D4").BorderAround _
      ColorIndex:=3, Weight:=xlThick
```

ChangeLink Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthChangeLinkC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthChangeLinkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthChangeLinkA "}

Changes a link from one document to another.

Syntax

expression.**ChangeLink(*Name, NewName, Type*)**

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

Name Required **String**. The name of the Microsoft Excel or DDE/OLE link to be changed, as it was returned from the **LinkSources** method.

NewName Required **String**. The new name of the link.

Type Optional Variant. The link type. Can be one of the following **XlLinkType** constants: **xlLinkTypeExcelLinks** or **xlLinkTypeOLELinks**. The default value is **xlLinkTypeExcelLinks**. Use **xlLinkTypeOLELinks** for both DDE and OLE links.

ChangeLink Method Example

This example changes a Microsoft Excel link.

```
ActiveWorkbook.ChangeLink "c:\excel\book1.xls", _
   "c:\excel\book2.xls", xlExcelLinks
```

ChartWizard Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthChartWizardC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthChartWizardX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthChartWizardA "}

Modifies the properties of the given chart.You can use this method toquickly format a chart without setting all the individual properties. This method is noninteractive, and it changes only the specified properties.

Syntax

expression.**ChartWizard(*Source, Gallery, Format, PlotBy, CategoryLabels, SeriesLabels, HasLegend, Title, CategoryTitle, ValueTitle, ExtraTitle*)**

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

Source Optional **Variant**. The range that contains the source data for the new chart. If this argument is omitted, Microsoft Excel edits the active chart sheet or the selected chart on the active worksheet.

Gallery Optional **Variant**. The chart type. Can be one of the following **XlChartType** constants: **xlArea**, **xlBar**, **xlColumn**, **xlLine**, **xlPie**, **xlRadar**, **xlXYScatter**, **xlCombination**, **xl3DArea**, **xl3DBar**, **xl3DColumn**, **xl3DLine**, **xl3DPie**, **xl3DSurface**, **xlDoughnut**, or **xlDefaultAutoFormat**.

Format Optional **Variant**. The option number for the built-in autoformats. Can be a number from 1 through 10, depending on the gallery type. If this argument is omitted, Microsoft Excel chooses a default value based on the gallery type and data source.

PlotBy Optional **Variant**. Specifies whether the data for each series is in rows or columns. Can be one of the following **XlRowCol** constants: **xlRows** or **xlColumns**.

CategoryLabels Optional **Variant**. An integer specifying the number of rows or columns within the source range that contain category labels. Legal values are from 0 (zero) through one less than the maximum number of the corresponding categories or series.

SeriesLabels Optional **Variant**. An integer specifying the number of rows or columns within the source range that contain series labels. Legal values are from 0 (zero) through one less than the maximum number of the corresponding categories or series.

HasLegend Optional **Variant**. **True** to include a legend.

Title Optional **Variant**. The chart title text.

CategoryTitle Optional **Variant**. The category axis title text.

ValueTitle Optional **Variant**. The value axis title text.

ExtraTitle Optional **Variant**. The series axis title for 3-D charts or the second value axis title for 2-D charts.

Remarks

If ***Source*** is omitted and either the selection isn't an embedded chart on the active worksheet or the active sheet isn't an existing chart, this method fails and an error occurs.

ChartWizard Method Example

This example reformats Chart1 as a line chart, adds a legend, and adds category and value axis titles.

```
Charts("Chart1").ChartWizard _
   Gallery:=xlLine, _
   HasLegend:=True, CategoryTitle:="Year", ValueTitle:="Sales"
```

ClearOutline Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearOutlineC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearOutlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearOutlineA "}

Clears the outline for the specified range.

Syntax

expression.**ClearOutline**

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

ClearOutline Method Example

This example clears the outline for the range A1:G37 on Sheet1.

```
Worksheets("Sheet1").Range("A1:G37").ClearOutline
```

ConsolidationFunction Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproConsolidationFunctionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConsolidationFunctionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConsolidationFunctionA "}

Returns the function code used for the current consolidation. Can be one of the following **XlConsolidationFunction** constants: **xlAverage**, **xlCount**, **xlCountNums**, **xlMax**, **xlMin**, **xlProduct**, **xlStDev**, **xlStDevP**, **xlSum**, **xlVar**, or **xlVarP**. Read-only **Long**.

ConsolidationFunction Property Example

This example displays a message box if the current consolidation is using the SUM function.

```
If Worksheets("Sheet1").ConsolidationFunction = xlSum Then
   MsgBox "Sheet1 uses the SUM function for consolidation."
End If
```

ConsolidationOptions Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproConsolidationOptionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConsolidationOptionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConsolidationOptionsA "}

Returns a three-element array of consolidation options, as shown in the following table. If the element is **True**, that option is set. Read-only **Variant**.

Element	Meaning
1	Use labels in top row
2	Use labels in left column
3	Create links to source data

ConsolidationOptions Property Example

This example displays the consolidation options for Sheet1. The list appears on a new worksheet created by the example.

```
Set newSheet = Worksheets.Add
aOptions = Worksheets("Sheet1").ConsolidationOptions
newSheet.Range("A1").Value = "Use labels in top row"
newSheet.Range("A2").Value = "Use labels in left column"
newSheet.Range("A3").Value = "Create links to source data"
For i = 1 To 3
   If aOptions(i) = True Then
      newSheet.Cells(i, 2).Value = "True"
   Else
      newSheet.Cells(i, 2).Value = "False"
   End If
Next i
newSheet.Columns("A:B").AutoFit
```

ConsolidationSources Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproConsolidationSourcesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConsolidationSourcesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConsolidationSourcesA "}

Returns an array of string values that name the source sheets for the worksheet's current consolidation. Returns **Empty** if there's no consolidation on the sheet. Read-only **Variant**.

ConsolidationSources Property Example

This example displays the names of the source ranges for the consolidation on Sheet1. The list appears on a new worksheet created by the example.

```
Set newSheet = Worksheets.Add
newSheet.Range("A1").Value = "Consolidation Sources"
aSources = Worksheets("Sheet1").ConsolidationSources
If IsEmpty(aSources) Then
   newSheet.Range("A2").Value = "none"
Else
   For i = 1 To UBound(aSources)
      newSheet.Cells(i + 1, 1).Value = aSources(i)
   Next i
End If
newSheet.Columns("A:B").AutoFit
```

DeleteChartAutoFormat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDeleteChartAutoFormatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDeleteChartAutoFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDeleteChartAutoFormatA "}

Removes a custom chart autoformat from the list of available chart autoformats.

Syntax

expression.**DeleteChartAutoFormat(*Name*)**

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

Name Required **String**. The name of the custom autoformat to be removed.

DeleteChartAutoFormat Method Example

This example deletes the custom autoformat named "Presentation Chart."

```
Application.DeleteChartAutoFormat name:="Presentation Chart"
```

DisplayClipboardWindow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayClipboardWindowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayClipboardWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayClipboardWindowA "}

Macintosh only. **True** if the Clipboard window is displayed. Read/write **Boolean**.

Remarks

In Windows, this property retains its value but does nothing.

DisplayClipboardWindow Property Example

This example displays the Clipboard window.

```
Application.DisplayClipboardWindow = True
```

DisplayRightToLeft Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayRightToLeftC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayRightToLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayRightToLeftA "}

True if the window displays from right to left instead of from left to right. Available only in the Arabic and Hebrew versions of Microsoft Excel. Read/write **Boolean**.

DisplayRightToLeft Property Example

This example sets window one to display from right to left.

```
ActiveWorkbook.Windows(1).DisplayRightToLeft = True
```

EditionOptions Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthEditionOptionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthEditionOptionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthEditionOptionsA "}

Sets options for publishers and subscribers in the workbook. Available only on the Macintosh running System 7.

Syntax

expression.**EditionOptions(*Type, Option, Name, Reference, Appearance, ChartSize, Formats*)**

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

Type Required **Long**. The edition type to be changed. Can be one of the following **XlEditionType** constants: **xlPublisher** or **xlSubscriber**.

Option Required **Long**. The type of information to set for the edition. If ***Type*** is **xlPublisher**, ***Option*** can be one of the following **XlEditionOptionsOption** constants: **xlCancel**, **xlSendPublisher**, **xlSelect**, **xlAutomaticUpdate**, **xlManualUpdate**, or **xlChangeAttributes**. If ***Type*** is **xlSubscriber**, ***Option*** can be one of the following **XlEditionOptionsOption** constants: **xlCancel**, **xlUpdateSubscriber**, **xlOpenSource**, **xlAutomaticUpdate**, or **xlManualUpdate**.

Name Optional **Variant**. The name of the edition, as returned from the **LinkSources** method. If ***Name*** is omitted, ***Reference*** must be specified.

Reference Optional **Variant** (required if ***Name*** isn't specified). The edition reference, as text in R1C1 style. This argument is required if there's more than one publisher or subscriber with the same edition name in the workbook, or if the ***Name*** argument isn't specified.

Appearance Optional **Variant**. If ***Option*** is **xlChangeAttributes**, this argument specifies whether the edition is published as shown on screen or as shown when printed. Can be one of the following **XlPictureAppearance** constants: **xlPrinter** or **xlScreen**.

ChartSize Optional **Variant**. If ***Option*** is **xlChangeAttributes** and the published object is a chart, this argument specifies the size of the edition. Can be one of the following **XlPictureAppearance** constants: **xlPrinter** or **xlScreen**. If the edition isn't a chart, this argument isn'tused.

Formats Optional **Variant**. If ***Option*** is **xlChangeAttributes**, this argument specifies the format of the published edition. Can be any combination of the following **XlEditionFormat** constants: **xlPICT**, **xlBIFF**, **xlRTF**, or **xlVALU**.

Extend Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthExtendC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthExtendX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthExtendA "}

Adds new data points to an existing series collection.

Syntax

expression.**Extend(*Source, Rowcol, CategoryLabels*)**

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

Source Required **Variant**. The new data to be added to the **SeriesCollection** object, either as a **Range** object or an array of data points.

Rowcol Optional **Variant**. Ignored if ***Source*** is an array. Specifies whether the new values are in the rows or columns of the given range source. Can be one of the following **XlRowCol** constants: **xlRows** or **xlColumns**. If this argument is omitted, Microsoft Excel attempts to determine where the values are by the size and orientation of the selected range or by the dimensions of the array.

CategoryLabels Optional **Variant**. Ignored if ***Source*** is an array. **True** to have the the first row or column contain the name of the category labels. **False** to have the first row or column contain the first data point of the series. If this argument is omitted, Microsoft Excel attempts to determine the location of the category label from the contents of the first row or column.

Extend Method Example

This example extends the series on Chart1 by adding the data in cells B1:B6 on Sheet1.

```
Charts("Chart1").SeriesCollection.Extend _
      Source:=Worksheets("Sheet1").Range("B1:B6")
```

FillAcrossSheets Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFillAcrossSheetsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFillAcrossSheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFillAcrossSheetsA "}

Copies a range to the same area on all other worksheets in a collection.

Syntax

expression.**FillAcrossSheets(*Range, Type*)**

expression Required. An expression that returns a **Sheets** or **Worksheets** object.

Range Required **Range**. The range to fill on all the worksheets in the collection. The range must be from a worksheet within the collection.

Type Optional **Variant**. Specifies how to copy the range. Can be one of the following **XlFillWith** constants: **xlFillWithAll**, **xlFillWithContents**, or **xlFillWithFormulas**. The default value is **xlFillWithAll**.

FillAcrossSheets Method Example

This example fills the range A1:C5 on Sheet1, Sheet5, and Sheet7 with the contents of the same range on Sheet1.

```
x = Array("Sheet1", "Sheet5", "Sheet7")
Sheets(x).FillAcrossSheets _
    Worksheets("Sheet1").Range("A1:C5")
```

LegendEntries Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthLegendEntriesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthLegendEntriesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthLegendEntriesA "}

Returns an object that represents either a single legend entry (a **LegendEntry** object, Syntax 1) or a collection of legend entries (a **LegendEntries** object, Syntax 2) for the legend.

Syntax 1

expression.**LegendEntries(*Index*)**

Syntax 2

expression.**LegendEntries**

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

Index Optional **Variant**. The number of the legend entry.

LegendEntries Method Example

This example sets the font for legend entry one on Chart1.

```
Charts("Chart1").Legend.LegendEntries(1).Font.Name = "Arial"
```

LegendKey Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLegendKeyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLegendKeyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLegendKeyA "}

Returns a **LegendKey** object that represents the legend key associated with the entry.

LegendKey Property Example

This example sets the legend key for legend entry one on Chart1 to be a triangle. The example should be run on a 2-D line chart.

```
Charts("Chart1").Legend.LegendEntries(1).LegendKey _
    .MarkerStyle = xlMarkerStyleTriangle
```

LinkInfo Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthLinkInfoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthLinkInfoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthLinkInfoA "}

Returns the link date and update status.

Syntax

expression.**LinkInfo(*Name, LinkInfo, Type, EditionRef*)**

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

Name Required **String**. The name of the link, as returned from the **LinkSources** method.

LinkInfo Required **Long**. The type of information to be returned. Can be one of the following **XlLinkInfo** constants: **xlUpdateState** or **xlEditionDate**. **xlEditionDate** applies only to editions. For **xlUpdateState**, this method returns 1 if the link updates automatically, or it returns 2 if the link must be updated manually.

Type Optional **Variant**. The type of link to return. Can be one of the following **XlLinkInfoType** constants: **xlLinkInfoOLELinks** (also handles DDE links), **xlLinkInfoPublishers**, or **xlLinkInfoSubscribers**.

EditionRef Optional **Variant**. If the link is an edition, this argument specifies the edition reference as a string in R1C1 style. This argument is required if there's more than one publisher or subscriber with the same name in the workbook.

LinkInfo Method Example

This example displays a message box if the link is updated automatically.

```
If ActiveWorkbook.LinkInfo( _
      "Word.Document|Document1!'!DDE_LINK1", xlUpdateState, _
         xlOLELinks) = 1 Then
   MsgBox "Link updates automatically"
End If
```

LinkSources Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthLinkSourcesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthLinkSourcesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthLinkSourcesA "}

Returns an array of links in the workbook. The names in the array are the names of the linked documents, editions, or DDE or OLE servers. Returns **Empty** if there are no links.

Syntax

expression.**LinkSources(*Type*)**

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

Type Optional. The type of link to return. Can be one of the following XlLink constants: **xlExcelLinks**, **xlOLELinks** (also handles DDE links), **xlPublishers**, or **xlSubscribers**.

Remarks

The format of the array is a one-dimensional array for all types but publisher and subscriber. The returned strings contain the name of the link source, in the appropriate notation for the link type. For example, DDE links use the "Server|Document!Item" syntax.

For publisher and subscriber links, the returned array is two-dimensional. The first column of the array contains the names of the edition, and the second column contains the references of the editions as text.

LinkSources Method Example

This example displays a list of OLE and DDE links in the active workbook. The example should be run on a workbook that contains one or more linked Word objects.

```
aLinks = ActiveWorkbook.LinkSources(xlOLELinks)
If Not IsEmpty(aLinks) Then
   For i = 1 To UBound(aLinks)
      MsgBox "Link " & i & ":" & Chr(13) & aLinks(i)
   Next i
End If
```

NameLocal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNameLocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNameLocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNameLocalA "}

Returns or sets the name of the object, in the language of the user. Read/write **String** for **Name**, read-only **String** for **Style**.

Remarks

If the style is a built-in style, this property returns the name of the style in the language of the current locale.

NameLocal Property Example

This example displays the name and localized name of style one in the active workbook.

```
With ActiveWorkbook.Styles(1)
   MsgBox "The name of the style is " & .Name
   MsgBox "The localized name of the style is " & .NameLocal
End With
```

NumberFormatLocal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNumberFormatLocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNumberFormatLocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNumberFormatLocalA "}

Returns or sets the format code for the object as a string in the language of the user. Read/write **String**.

Remarks

The **Format** function uses different format code strings than do the **NumberFormat** and **NumberFormatLocal** properties.

NumberFormatLocal Property Example

This example displays the number format for cell A1 on Sheet1, in the language of the user.

```
MsgBox "The number format for cell A1 is " & _
   Worksheets("Sheet1").Range("A1").NumberFormatLocal
```

Object Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproObjectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproObjectA "}

Returns the OLE Automation object associated with this OLE object. Read-only **Object**.

Object Property Example

This example inserts text at the beginning of an embedded Word document object on Sheet1. Note that the three statements in the **With** control structure are WordBasic statements.

```
Set wordObj = Worksheets("Sheet1").OLEObjects(1)
wordObj.Activate
With wordObj.Object.Application.WordBasic
    .StartOfDocument
    .Insert "This is the beginning"
    .InsertPara
End With
```

OLEObjects Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOLEObjectsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOLEObjectsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOLEObjectsA "}

Returns an object that represents either a single OLE object (an **OLEObject**, Syntax 1) or a collection of all OLE objects (an **OLEObjects** collection, Syntax 2) on the chart or sheet. Read-only.

Syntax 1

expression.**OLEObjects(*Index*)**

Syntax 2

expression.**OLEObjects**

expression Required. An expression that returns a **Chart** or **Worksheet** object.

Index Optional **Variant**. The name or number of the OLE object.

OLEObjects Method Example

This example creates a list of link types for OLE objects on Sheet1. The list appears on a new worksheet created by the example.

```
Set newSheet = Worksheets.Add
i = 2
newSheet.Range("A1").Value = "Name"
newSheet.Range("B1").Value = "Link Type"
For Each obj In Worksheets("Sheet1").OLEObjects
   newSheet.Cells(i, 1).Value = obj.Name
   If obj.OLEType = xlOLELink Then
      newSheet.Cells(i, 2) = "Linked"
   Else
      newSheet.Cells(i, 2) = "Embedded"
   End If
   i = i + 1
Next
```

OLEType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOLETypeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOLETypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOLETypeA "}

Returns the OLE object type. Can be one of the following **XlOLEType** constants: **xlOLELink** or **xlOLEEmbed**. Returns **xlOLELink** if the object is linked (it exists outside of the file), or returns **xlOLEEmbed** if the object is embedded (it's entirely contained within the file). Read-only **Long**.

OLEType Property Example

This example creates a list of link types for OLE objects on Sheet1. The list appears on a new worksheet created by the example.

```
Set newSheet = Worksheets.Add
i = 2
newSheet.Range("A1").Value = "Name"
newSheet.Range("B1").Value = "Link Type"
For Each obj In Worksheets("Sheet1").OLEObjects
   newSheet.Cells(i, 1).Value = obj.Name
   If obj.OLEType = xlOLELink Then
      newSheet.Cells(i, 2) = "Linked"
   Else
      newSheet.Cells(i, 2) = "Embedded"
   End If
   i = i + 1
Next
```

OpenLinks Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOpenLinksC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOpenLinksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOpenLinksA "}

Opens the supporting documents for a link or links.

Syntax

expression.**OpenLinks(*Name, ReadOnly, Type*)**

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

Name Required **String**. The name of the Microsoft Excel or DDE/OLE link, as returned from the **LinkSources** method.

ReadOnly Optional Variant. **True** to open documents as read-only. The default value is **False**.

Type Optional. The link type. Can be one of the following **XlLink** constants: **xlExcelLinks**, **xlOLELinks** (also handles DDE links), **xlPublishers**, or **xlSubscribers**.

OpenLinks Method Example

This example opens OLE link one in the active workbook.

```
linkArray = ActiveWorkbook.LinkSources(xlOLELinks)
ActiveWorkbook.OpenLinks linkArray(1)
```

This example opens all supporting Microsoft Excel documents for the active workbook.

```
ActiveWorkbook.OpenLinks _
   name:=ActiveWorkbook.LinkSources(xlExcelLinks)
```

OutlineLevel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOutlineLevelC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOutlineLevelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOutlineLevelA "}

Returns or sets the current outline level of the specified row or column. Read/write **Variant**.

Remarks

Level one is the outermost summary level.

OutlineLevel Property Example

This example sets the outline level for row two on Sheet1.

```
Worksheets("Sheet1").Rows(2).OutlineLevel = 1
```

Parse Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthParseC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthParseX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthParseA "}

Parses a range of data and breaks it into multiple cells. Distributes the contents of the range to fill several adjacent columns; the range can be no more than one column wide.

Syntax

expression.**Parse(*ParseLine, Destination*)**

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

ParseLine Optional **Variant**. A string that contains left and right brackets to indicate where the cells should be split. For example, "`[xxx][xxx]`" would insert the first three characters into the first column of the destination range, and it would insert the next three characters into the second column. If this argument is omitted, Microsoft Excel guesses where to split the columns based on the spacing of the top left cell in the range. If you want to use a different range to guess the parse line, use a **Range** object as the ***ParseLine*** argument. That range must be one of the cells that's being parsed. The ***ParseLine*** argument cannot be longer than 255 characters, including the brackets and spaces.

Destination Optional **Variant**. A **Range** object that represents the upper-left corner of the destination range for the parsed data. If this argument is omitted, Microsoft Excel parses in place.

Parse Method Example

This example divides telephone numbers of the form 206-555-1212 into two columns. The first column contains only the area code, and the second column contains the seven-digit telephone number with the embedded hyphen.

```
Worksheets("Sheet1").Columns("A").Parse _
   parseLine:="[xxx] [xxxxxxxx]", _
   destination:=Worksheets("Sheet1").Range("B1")
```

Period Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPeriodC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPeriodX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPeriodA "}

Returns or sets the period for the moving-average trendline. Read/write **Long**.

Period Property Example

This example sets the period for the moving-average trendline on Chart1. The example should be run on a 2-D column chart with a single series that contains 10 data points and a moving-average trendline.

```
With Charts("Chart1").SeriesCollection(1).Trendlines(1)
   If .Type = xlMovingAvg Then .Period = 5
End With
```

PlotVisibleOnly Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPlotVisibleOnlyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPlotVisibleOnlyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPlotVisibleOnlyA "}

True if only visible cells are plotted. **False** if both visible and hidden cells are plotted. Read/write **Boolean**.

PlotVisibleOnly Property Example

This example causes Microsoft Excel to plot only visible cells in Chart1.

```
Charts("Chart1").PlotVisibleOnly = True
```

Precedents Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrecedentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrecedentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrecedentsA "}

Returns a **Range** object that represents all the precedents of a cell. This can be a multiple selection (a union of **Range** objects) if there's more than one precedent. Read-only.

Precedents Property Example

This example selects the precedents of cell A1 on Sheet1.

```
Worksheets("Sheet1").Activate
Range("A1").Precedents.Select
```

PrefixCharacter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrefixCharacterC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrefixCharacterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrefixCharacterA "}

Returns the prefix character for the cell. Read-only **Variant**.

Remarks

If the **TransitionNavigKeys** property is **False**, this prefix character will be ' for a text label, or blank. If the **TransitionNavigKeys** property is **True**, this character will be ' for a left-justified label, " for a right-justified label, ^ for a centered label, \ for a repeated label, or blank.

PrefixCharacter Property Example

This example displays the prefix character for cell A1 on Sheet1.

```
MsgBox "The prefix character is " & _
   Worksheets("Sheet1").Range("A1").PrefixCharacter
```

PrintArea Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintAreaC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintAreaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintAreaA "}

Returns or sets the range to be printed, as a string using A1-style references in the language of the macro. Read/write **String**.

Remarks

Set this property to **False** or to the empty string ("") to set the print area to the entire sheet.

This property applies only to worksheet pages.

PrintArea Property Example

This example sets the print area to cells A1:C5 on Sheet1.

```
Worksheets("Sheet1").PageSetup.PrintArea = "$A$1:$C$5"
```

This example sets the print area to the current region on Sheet1. Note that you use the **Address** property to return an A1-style address.

```
Worksheets("Sheet1").Activate
ActiveSheet.PageSetup.PrintArea = _
   ActiveCell.CurrentRegion.Address
```

PrintTitleColumns Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintTitleColumnsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintTitleColumnsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintTitleColumnsA "}

Returns or sets the columns that contain the cells to be repeated on the left side of each page, as a string in A1-style notation in the language of the macro. Read/write **String**.

Remarks

If you specify only part of a column or columns, Microsoft Excel expands the range to full columns.

Set this property to **False** or to the empty string ("") to turn off title columns.

This property applies only to worksheet pages.

PrintTitleColumns Property Example

This example defines row three as the title row, and it defines columns one through three as the title columns.

```
Worksheets("Sheet1").Activate
ActiveSheet.PageSetup.PrintTitleRows = ActiveSheet.Rows(3).Address
ActiveSheet.PageSetup.PrintTitleColumns = _
      ActiveSheet.Columns("A:C").Address
```

PrintTitleRows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintTitleRowsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintTitleRowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintTitleRowsA "}

Returns or sets the rows that contain the cells to be repeated at the top of each page, as a string in A1-style notation in the language of the macro. Read/write **String**.

Remarks

If you specify only part of a row or rows, Microsoft Excel expands the range to full rows.

Set this property to **False** or to the empty string ("") to turn off title rows.

This property applies only to worksheet pages.

PrintTitleRows Property Example

This example defines row three as the title row, and it defines columns one through three as the title columns.

```
Worksheets("Sheet1").Activate
ActiveSheet.PageSetup.PrintTitleRows = ActiveSheet.Rows(3).Address
ActiveSheet.PageSetup.PrintTitleColumns = _
      ActiveSheet.Columns("A:C").Address
```

ProtectScenarios Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectScenariosC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectScenariosX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectScenariosA "}

True if the worksheet scenarios are protected. Read-only **Boolean**.

ProtectScenarios Property Example

This example displays a message box if scenarios are protected on Sheet1.

```
If Worksheets("Sheet1").ProtectScenarios Then _
   MsgBox "Scenarios are protected on this worksheet."
```

RefersTo Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefersToC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefersToX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefersToA "}

Returns or sets the formula that the name is defined to refer to, in the language of the macro and in A1-style notation, beginning with an equal sign. Read/write **String**.

RefersTo Property Example

This example creates a list of all the names in the active workbook, and it shows their formulas in A1-style notation in the language of the macro. The list appears on a new worksheet created by the example.

```
Set newSheet = Worksheets.Add
i = 1
For Each nm In ActiveWorkbook.Names
   newSheet.Cells(i, 1).Value = nm.Name
   newSheet.Cells(i, 2).Value = "'" & nm.RefersTo
   i = i + 1
Next
newSheet.Columns("A:B").AutoFit
```

ShortcutKey Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShortcutKeyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShortcutKeyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShortcutKeyA "}

Returns or sets the shortcut key for a name defined as a custom Microsoft Excel 4.0 macro command. Read/write **String**.

ShortcutKey Property Example

This example sets the shortcut key for name one in the active workbook. The example should be run on a workbook in which name one refers to a Microsoft Excel 4.0 command macro.

```
ActiveWorkbook.Names(1).ShortcutKey = "K"
```

ShowAllData Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowAllDataC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowAllDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowAllDataA "}

Makes all rows of the currently filtered list visible. If AutoFilter is in use, this method changes the arrows to "All."

Syntax

expression.**ShowAllData**

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

ShowAllData Method Example

This example makes all data on Sheet1 visible. The example should be run on a worksheet that contains a list you filtered using the **AutoFilter** command.

```
Worksheets("Sheet1").ShowAllData
```

ShowToolTips Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowToolTipsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowToolTipsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowToolTipsA "}

True if ToolTips are turned on. Read/write **Boolean**.

ShowToolTips Property Example

This example causes Microsoft Excel to display ToolTips.

```
Application.ShowToolTips = True
```

SizeWithWindow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSizeWithWindowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSizeWithWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSizeWithWindowA "}

True if Microsoft Excel resizes the chart to match the size of the chart sheet window. **False** if the chart size isn't attached to the window size. Applies only to chart sheets (doesn't apply to embedded charts). Read/write **Boolean**.

SizeWithWindow Property Example

This example sets Chart1 to be sized to its window.

```
Charts("Chart1").SizeWithWindow = True
```

TextToColumns Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthTextToColumnsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthTextToColumnsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthTextToColumnsA "}

Parses a column of cells that contain text into several columns.

Syntax

expression.**TextToColumns(*Destination, DataType, TextQualifier, ConsecutiveDelimiter, Tab, Semicolon, Comma, Space, Other, OtherChar, FieldInfo*)**

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

Destination Optional **Variant**. A **Range** object that specifies where Microsoft Excel will place the results. If the range is larger than a single cell, the top left cell is used.

DataType Optional **Variant**. The format of the text to be split into columns. Can be one of the following **XlTextParsingType** constants: **xlDelimited** or **xlFixedWidth**. The default value is **xlDelimited**.

TextQualifier Optional **Variant**. The text qualifier. Can be one of the following **XlTextQualifier** constants: **xlTextQualifierDoubleQuote**, **xlTextQualifierSingleQuote**, or **xlTextQualifierNone**. The default value is **xlTextQualifierDoubleQuote**.

ConsecutiveDelimiter Optional **Variant**. **True** to have Microsoft Excel consider consecutive delimiters as one delimiter. The default value is **False**.

Tab Optional **Variant**. **True** to have ***DataType*** be **xlDelimited** and to have the tab character be a delimiter. The default value is **False**.

Semicolon Optional **Variant**. **True** to have ***DataType*** be **xlDelimited** and to have the semicolon be a delimiter. The default value is **False**.

Comma Optional **Variant**. **True** to have ***DataType*** be **xlDelimited** and to have the comma be a delimiter. The default value is **False**.

Space Optional **Variant**. **True** to have ***DataType*** be **xlDelimited** and to have the space character be a delimiter. The default value is **False**.

Other Optional **Variant**. **True** to have ***DataType*** be **xlDelimited** and to have the character be specified by the ***OtherChar*** argument be a delimiter. The default value is **False**.

OtherChar Optional **Variant**. (required if ***Other*** is **True**). The delimiter character when ***Other*** is **True**. If more than one character is specified, only the first character of the string is used; the remaining characters are ignored.

FieldInfo Optional **Variant**. An array containing parse information for the individual columns of data. The interpretation depends on the value of ***DataType***.

When the data is delimited, this argument is an array of two-element arrays, with each two-element array specifying the conversion options for a particular column. The first element is the column number (1-based), and the second element is one of the following numbers, specifying how the column is parsed:

1	General
2	Text
3	MDY date
4	DMY date
5	YMD date
6	MYD date
7	DYM date
8	YDM date
9	Skip the column

The column specifiers can be in any order. If a given column specifier is not present for a particular

column in the input data, the column is parsed with the **General** setting. This example causes the third column to be skipped, the first column to be parsed as text, and the remaining columns in the source data to be parsed with the **General** setting.

```
Array(Array(3, 9), Array(1, 2))
```

If the source data has fixed-width columns, the first element of each two-element array specifies the starting character position in the column (as an integer; 0 (zero) is the first character). The second element of the two-element array specifies the parse option for the column as a number from 1 through 9, as listed above.

The following example parses two columns from a fixed-width file, with the first column starting at the beginning of the line and extending for 10 characters. The second column starts at position 15 and goes to the end of the line. To avoid including the characters between position 10 and position 15, Microsoft Excel adds a skipped column entry.

```
Array(Array(0, 1), Array(10, 9), Array(15, 1))
```

TextToColumns Method Example

This example converts the contents of the Clipboard, which contains a space-delimited text table, into separate columns on Sheet1. You can create a simple space-delimited table in Notepad or WordPad (or another text editor), copy the text table to the Clipboard, switch to Microsoft Excel, and then run this example.

```
Worksheets("Sheet1").Activate
ActiveSheet.Paste
Selection.TextToColumns DataType:=xlDelimited, _
   ConsecutiveDelimiter:=True, Space:=True
```

TransitionMenuKeyAction Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTransitionMenuKeyActionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransitionMenuKeyActionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransitionMenuKeyActionA "}

Returns or sets the action taken when the alternate menu key is pressed. Can be either **xlExcelMenus** or **xlLotusHelp**. This property cannot be set to **xlLotusHelp** on the Macintosh. Read/write **Long**.

TransitionMenuKeyAction Property Example

This example sets the alternate menu key to run Lotus 1-2-3 Help when it's pressed. This property cannot be set to **xlLotusHelp** on the Macintosh.

```
Application.TransitionMenuKeyAction = xlLotusHelp
```

Update Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUpdateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUpdateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUpdateA "}

Updates the link or PivotTable.

Syntax

expression.**Update**

expression Required. An expression that returns an **OLEObject** or **PivotTable** object.

Update Method Example

This example updates the link to OLE object one on Sheet1.

```
Worksheets("Sheet1").OLEObjects(1).Update
```

UpdateLink Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUpdateLinkC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUpdateLinkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUpdateLinkA "}

Updates a Microsoft Excel, DDE, or OLE link (or links).

Syntax

expression.**UpdateLink(*Name, Type*)**

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

Name Required **String**. The name of the Microsoft Excel or DDE/OLE link to be updated, as returned from the **LinkSources** method.

Type Optional **Variant**. The link type. Can be one of the following **XlLinkType** constants: **xlLinkTypeExcelLinks** or **xlLinkTypeOLELinks** (also used for DDE links). The default value is **xlLinkTypeExcelLinks**.

UpdateLink Method Example

This example updates all links in the active workbook.

```
ActiveWorkbook.UpdateLink Name:=ActiveWorkbook.LinkSources
```

Verb Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthVerbC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthVerbX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthVerbA "}

Sends a verb to the server of the specified OLE object.

Syntax

expression.**Verb(*Verb*)**

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

Verb Optional **Variant**. The verb that the server of the OLE object should act on. If this argument is omitted, the default verb is sent. The available verbs are determined by the object's source application. Typical verbs for an OLE object are Open and Primary (represented by the **XlOLEVerb** constants **xlOpen** and **xlPrimary**).

Verb Method Example

This example sends the default verb to the server for OLE object one on Sheet1.

```
Worksheets("Sheet1").OLEObjects(1).Verb
```

VisibleRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVisibleRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVisibleRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVisibleRangeA "}

Returns a **Range** object that represents the range of cells that are visible in the window or pane. If a column or row is partially visible, it's included in the range. Read-only.

VisibleRange Property Example

This example displays the number of cells visible on Sheet1.

```
Worksheets("Sheet1").Activate
MsgBox "There are " & Windows(1).VisibleRange.Cells.Count _
   & " cells visible"
```

WallsAndGridlines2D Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWallsAndGridlines2DC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWallsAndGridlines2DX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWallsAndGridlines2DA "}

True if gridlines are drawn two-dimensionally on a 3-D chart. Read/write **Boolean**.

WallsAndGridlines2D Property Example

This example causes Microsoft Excel to draw 2-D gridlines on Chart1.

```
Charts("Chart1").WallsAndGridlines2D = True
```

Paste Method (Chart Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPasteChartObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPasteChartObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPasteChartObjA "}

Pastes chart data from the Clipboard into the specified chart.

Syntax

expression.**Paste(*Type*)**

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

Type Optional **Variant**. Specifies the chart information to paste if a chart is on the Clipboard. Can be one of the following **XlPasteType** constants: **xlFormats**, **xlFormulas**, or **xlAll**. The default value is **xlAll**. If there's data other than a chart on the Clipboard, this argument cannot be used.

Remarks

This method changes the current selection.

Paste Method (Chart Object) Example

This example pastes data from the range B1:B5 on Sheet1 into Chart1.

```
Worksheets("Sheet1").Range("B1:B5").Copy
Charts("Chart1").Paste
```

Paste Method (Point or Series Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPastePointObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPastePointObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPastePointObjA "}

Pastes a picture from the Clipboard as the marker on the selected point or series. This method can be used on column, bar, line, or radar charts, and it sets the **MarkerStyle** property to **xlMarkerStylePicture**.

Syntax

expression.**Paste**

expression Required. An expression that returns a **Point** or **Series** object.

Paste Method (Point or Series Object) Example

This example pastes a picture from the Clipboard into series one in Chart1.

```
Charts("Chart1").SeriesCollection(1).Paste
```

Paste Method (SeriesCollection Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPasteSeriesCollectionObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPasteSeriesCollectionObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPasteSeriesCollectionObjA "}

Pastes data from the Clipboard into the specified series collection.

Syntax

expression.**Paste**(***Rowcol, SeriesLabels, CategoryLabels, Replace, NewSeries***)

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

Rowcol Optional **Variant**. Specifies whether the values corresponding to a particular data series are in rows or columns. Can be one of the following **XlRowCol** constants: **xlRows** or **xlColumns**. The default value is **xlColumns**.

SeriesLabels Optional **Variant**. **True** to use the contents of the cell in the first column of each row (or the first row of each column) as the name of the data series in that row (or column). **False** to use the contents of the cell in the first column of each row (or the first row of each column) as the first data point in the data series. The default value is **False**.

CategoryLabels Optional **Variant**. **True** to use the contents of the first row (or column) of the selection as the categories for the chart. **False** to use the contents of the first row (or column) as the first data series in the chart. The default value is **False**.

Replace Optional **Variant**. **True** to apply categories while replacing existing categories with information from the copied range. **False** to insert new categories without replacing any old ones. The default value is **True**.

NewSeries Optional **Variant**. **True** to paste the data as a new series. **False** to paste the data as new points in an existing series. The default value is **True**.

Paste Method (SeriesCollection Collection) Example

This example copies data to the Clipboard from cells C1:C5 on Sheet1 and then pastes the data into Chart1 as a new series.

```
Worksheets("Sheet1").Range("C1:C5").Copy
Charts("Chart1").SeriesCollection.Paste
```

Paste Method (Worksheet Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPasteWorksheetObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPasteWorksheetObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPasteWorksheetObjA "}

Pastes the contents of the Clipboard onto the sheet.

Syntax

expression.**Paste**(***Destination, Link***)

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

Destination Optional **Variant**. A **Range** object that specifies where the Clipboard contents should be pasted. If this argument is omitted, the current selection is used. This argument can be specified only if the contents of the Clipboard can be pasted into a range. If this argument is specified, the ***Link*** argument cannot be used.

Link Optional **Variant**. **True** to establish a link to the source of the pasted data. If this argument is specified, the ***Destination*** argument cannot be used. The default value is **False**.

Remarks

If you don't specify the ***Destination*** argument, you must select the destination range before you use this method.

This method may modify the sheet selection, depending on the contents of the Clipboard.

Paste Method (Worksheet Object) Example

This example copies data from cells C1:C5 on Sheet1 to cells D1:D5 on Sheet1.

```
Worksheets("Sheet1").Range("C1:C5").Copy
ActiveSheet.Paste Destination:=Worksheets("Sheet1").Range("D1:D5")
```

PasteSpecial Method

Range object: Pastes data from the Clipboard into the specified range.

Worksheet object: Pastes data from the Clipboard onto the sheet, using a specified format. Use this method to paste data from other applications or to paste data in a specific format.

PasteSpecial Method (Range Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPasteSpecialRangeObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPasteSpecialRangeObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPasteSpecialRangeObjA "}

Pastes a **Range** from the Clipboard into the specified range.

Syntax

expression.**PasteSpecial(*Paste, Operation, SkipBlanks, Transpose*)**

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

Paste Optional **Variant**. The part of the range to be pasted. Can be one of the following **XlPasteType** constants: **xlPasteAll**, **xlPasteFormulas**, **xlPasteValues**, **xlPasteFormats**, **xlPasteNotes**, or **xlPasteAllExceptBorders**. The default value is **xlPasteAll**.

Operation Optional **Variant**. The paste operation. Can be one of the following **XlPasteSpecialOperation** constants: **xlPasteSpecialOperationNone**, **xlPasteSpecialOperationAdd**, **xlPasteSpecialOperationSubtract**, **xlPasteSpecialOperationMultiply**, or **xlPasteSpecialOperationDivide**. The default value is **xlPasteSpecialOperationNone**.

SkipBlanks Optional **Variant**. **True** to have blank cells in the range on the Clipboard not be pasted into the destination range. The default value is **False**.

Transpose Optional **Variant**. **True** to transpose rows and columns when the range is pasted.The default value is **False**.

PasteSpecial Method (Range Object) Example

This example replaces the data in cells D1:D5 on Sheet1 with the sum of the existing contents and cells C1:C5 on Sheet1.

```
With Worksheets("Sheet1")
    .Range("C1:C5").Copy
    .Range("D1:D5").PasteSpecial Operation:=xlPasteSpecialOperationAdd
End With
```

PasteSpecial Method (Worksheet Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPasteSpecialWorksheetObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPasteSpecialWorksheetObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPasteSpecialWorksheetObjA "}

Pastes the contents of the Clipboard onto the sheet, using a specified format. Use this method to paste data from other applications or to paste data in a specific format.

Syntax

expression.**PasteSpecial(*Format, Link, DisplayAsIcon, IconFileName, IconIndex, IconLabel*)**

expression Required. An expression that returns a **DialogSheet** or **Worksheet** object.

Format Optional **Variant**. A string that specifies the Clipboard format of the data.

Link Optional **Variant**. **True** to establish a link to the source of the pasted data. If the source data isn't suitable for linking or the source application doesn't support linking, this parameter is ignored. The default value is **False**.

DisplayAsIcon Optional **Variant**. **True** to display the pasted as an icon. The default value is **False**.

IconFileName Optional **Variant**. The name of the file that contains the icon to use if ***DisplayAsIcon*** is **True**.

IconIndex Optional **Variant**. The index number of the icon within the icon file.

IconLabel Optional **Variant**. The text label of the icon.

Remarks

You must select the destination range before you use this method.

This method may modify the sheet selection, depending on the contents of the Clipboard.

PasteSpecial Method (Worksheet Object) Example

This example pastes a Microsoft Word document object from the Clipboard to cell D1 on Sheet1.

```
Worksheets("Sheet1").Range("D1").Select
ActiveSheet.PasteSpecial format:="Microsoft Word 8.0 Document Object"
```

This example pastes the same Microsoft Word document object and displays it as an icon.

```
Worksheets("Sheet1").Range("F5").Select
ActiveSheet.PasteSpecial _
      Format:="Microsoft Word 8.0 Document Object", _
      DisplayAsIcon:=True
```

ApplyDataLabels Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthApplyDataLabelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthApplyDataLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthApplyDataLabelsA "}

Applies data labels to a point, a series, or all the series in a chart.

Syntax

expression.**ApplyDataLabels(*Type, LegendKey*)**

expression Required. An expression that returns a **Chart**, **Point**, or **Series** object.

Type Optional **Variant**. The data label type. Can be one of the following **XlDataLabelsType** constants.

Constant	Description
xlDataLabelsShowNone	No data labels.
xlDataLabelsShowValue	Value for the point (assumed if this argument isn't specified).
xlDataLabelsShowPercent	Percentage of the total. Available only for pie charts and doughnut charts.
xlDataLabelsShowLabel	Category for the point. This is the default value.
xlDataLabelsShowLabelAndPercent	Percentage of the total, and category for the point. Available only for pie charts and doughnut charts.

LegendKey Optional **Variant**. **True** to show the legend key next to the point. The default value is **False**.

ApplyDataLabels Method Example

This example applies category labels to series one in Chart1.

```
Charts("Chart1").SeriesCollection(1). _
    ApplyDataLabels Type:=xlDataLabelsShowLabel
```

DataLabel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataLabelC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataLabelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataLabelA "}

Returns a **DataLabel** object that represents the data label associated with the point or trendline. Read-only.

DataLabel Property Example

This example turns on the data label for point seven in series three in Chart1, and then it sets the data label color to blue.

```
With Charts("Chart1").SeriesCollection(3).Points(7)
    .HasDataLabel = True
    .ApplyDataLabels type:=xlValue
    .DataLabel.Font.ColorIndex = 5
End With
```

HasDataLabel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasDataLabelC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasDataLabelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasDataLabelA "}

True if the point has a data label. Read/write **Boolean**.

HasDataLabel Property Example

This example turns on the data label for point seven in series three in Chart1, and then it sets the data label color to blue.

```
With Charts("Chart1").SeriesCollection(3).Points(7)
    .HasDataLabel = True
    .ApplyDataLabels type:=xlValue
    .DataLabel.Font.ColorIndex = 5
End With
```

MarkerStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarkerStyleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarkerStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarkerStyleA "}

Returns or sets the marker style for a point or series in a line chart, scatter chart, or radar chart. Read/write **Long**.

Can be one of the following **XlMarkerStyle** constants.

Constant	Description
xlMarkerStyleNone	No markers
xlMarkerStyleAutomatic	Automatic markers
xlMarkerStyleSquare	Square markers
xlMarkerStyleDiamond	Diamond-shaped markers
xlMarkerStyleTriangle	Triangular markers
xlMarkerStyleX	Square markers with an X
xlMarkerStyleStar	Square markers with an asterisk
xlMarkerStyleDot	Short bar markers
xlMarkerStyleDash	Long bar markers
xlMarkerStyleCircle	Circular markers
xlMarkerStylePlus	Square markers with a plus sign
xlMarkerStylePicture	Picture markers

MarkerStyle Property Example

This example sets the marker style for series one in Chart1. The example should be run on a 2-D line chart.

```
Charts("Chart1").SeriesCollection(1).MarkerStyle = xlMarkerStyleCircle
```

ApplyNames Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthApplyNamesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthApplyNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthApplyNamesA "}

Applies names to the cells in the specified range.

Syntax

expression.**ApplyNames(*Names, IgnoreRelativeAbsolute, UseRowColumnNames, OmitColumn, OmitRow, Order, AppendLast*)**

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

Names Optional **Variant**. An array of the names to be applied. If this argument is omitted, all names on the sheet are applied to the range.

IgnoreRelativeAbsolute Optional **Variant**. **True** to replace references with names, regardless of the reference types of either the names or references. **False** to replace absolute references only with absolute names, relative references only with relative names, and mixed references only with mixed names. The default value is **True**.

UseRowColumnNames Optional **Variant**. **True** to use the names of row and column ranges that contain the specified range if names for the range cannot be found. **False** to ignore the ***OmitColumn*** and ***OmitRow*** arguments. The default value is **True**.

OmitColumn Optional **Variant**. **True** to replace the entire reference with the row-oriented name. The column-oriented name can be omitted only if the referenced cell is in the same column as the formula and is within a row-oriented named range. The default value is **True**.

OmitRow Optional **Variant**. **True** to replace the entire reference with the column-oriented name. The row-oriented name can be omitted only if the referenced cell is in the same row as the formula and is within a column-oriented named range. The default value is **True**.

Order Optional **Variant**. Determines which range name is listed first when a cell reference is replaced by a row-oriented and column-oriented range name. Can be one of the following **XlApplyNamesOrder** constants: **xlRowThenColumn** or **xlColumnThenRow**.

AppendLast Optional **Variant**. **True** to replace the definitions of the names in ***Names*** and also replace the definitions of the last names that were defined. **False** to replace the definitions of the names in ***Names*** only. The default value is **False**.

Remarks

You can use the **Array** function to create the list of names for the ***Names*** argument.

If you want to apply names to the entire sheet, use `Cells.ApplyNames`.

You cannot "unapply" names; to delete names, use the **Delete** method.

ApplyNames Method Example

This example applies names to the entire sheet.

```
Cells.ApplyNames Names:=Array("Sales", "Profits")
```

ApplyOutlineStyles Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthApplyOutlineStylesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthApplyOutlineStylesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthApplyOutlineStylesA "}

Applies outlining styles to the specified range.

Syntax

expression.**ApplyOutlineStyles**

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

ApplyOutlineStyles Method Example

The following example applies automatic outlining styles to the selection. The selection must include the entire outline range on a worksheet.

```
Selection.ApplyOutlineStyles
```

AutoFill Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoFillC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoFillX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoFillA "}

Performs an autofill on the cells in the specified range.

Syntax

expression.**AutoFill(*Destination, Type*)**

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

Destination Required **Range** object. The cells to be filled. The destination must include the source range.

Type Optional **Variant**. Specifies the fill type. Can be one of the following **XlFillType** constants: **xlFillDefault**, **xlFillSeries**, **xlFillCopy**, **xlFillFormats**, **xlFillValues**, **xlFillDays**, **xlFillWeekdays**, **xlFillMonths**, **xlFillYears**, **xlLinearTrend**, or **xlGrowthTrend**. If this argument is **xlFillDefault** or omitted, Microsoft Excel selects the most appropriate fill type, based on the source range.

AutoFill Method Example

This example performs an autofill on cells A1:A20 on Sheet1, based on the source range A1:A2 on Sheet1. Before running this example, type **1** in cell A1 and type **2** in cell A2.

```
Set sourceRange = Worksheets("Sheet1").Range("A1:A2")
Set fillRange = Worksheets("Sheet1").Range("A1:A20")
sourceRange.AutoFill Destination:=fillRange
```

AutoFormat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoFormatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoFormatA "}

Automatically formats a range of cells, using a predefined format.

Syntax

expression.**AutoFormat(*Format, Number, Font, Alignment, Border, Pattern, Width*)**

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

Format Optional **Variant**. The autoformat. Can be one of the following **XlRangeAutoFormat** constants:

xlRangeAutoFormat3DEffects1	**xlRangeAutoFormatColor3**
xlRangeAutoFormat3DEffects2	**xlRangeAutoFormatList1**
xlRangeAutoFormatAccounting1	**xlRangeAutoFormatList2**
xlRangeAutoFormatAccounting2	**xlRangeAutoFormatList3**
xlRangeAutoFormatAccounting3	**xlRangeAutoFormatLocalFormat1**
xlRangeAutoFormatAccounting4	**xlRangeAutoFormatLocalFormat2**
xlRangeAutoFormatClassic1	**xlRangeAutoFormatLocalFormat3**
xlRangeAutoFormatClassic2	**xlRangeAutoFormatLocalFormat4**
xlRangeAutoFormatClassic3	**xlRangeAutoFormatNone**
xlRangeAutoFormatColor1	**xlRangeAutoFormatSimple**
xlRangeAutoFormatColor2	

The default value is **xlRangeAutoFormatClassic1**. The **xlRangeAutoFormatLocalFormat1**, **xlRangeAutoFormatLocalFormat2**, **xlRangeAutoFormatLocalFormat3**, and **xlRangeAutoFormatLocalFormat4** constants are not used in U.S. English Microsoft Excel.

Number Optional **Variant**. **True** to include number formats in the autoformat. The default value is **True**.

Font Optional **Variant**. **True** to include font formats in the autoformat. The default value is **True**.

Alignment Optional **Variant**. **True** to include alignment in the autoformat. The default value is **True**.

Border Optional **Variant**. **True** to include border formats in the autoformat. The default value is **True**.

Pattern Optional **Variant**. **True** to include pattern formats in the autoformat. The default value is **True**.

Width Optional **Variant**. **True** to include column width and row height in the autoformat. The default value is **True**.

Remarks

If the range is a single cell, this method also formats the current region surrounding the cell. In other words, the following two statements are equivalent:

```
Cells("A1").AutoFormat
Cells("A1").CurrentRegion.AutoFormat
```

AutoFormat Method Example

This example formats cells A1:D8 on Sheet1, using a predefined format.

```
Worksheets("Sheet1").Range("A1:D8"). _
    AutoFormat Format:=xlRangeAutoFormatClassic1
```

AutoFit Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoFitC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoFitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoFitA "}

Changes the width of the columns in the range or the height of the rows in the range to achieve the best fit.

Syntax

expression.**AutoFit**

expression Required. An expression that returns a **Range** object. Must be a row or a range of rows, or a column or a range of columns. Otherwise, this method generates an error.

Remarks

One unit of column width is equal to the width of one character in the Normal style.

AutoFit Method Example

This example changes the width of columns A through I on Sheet1 to achieve the best fit.

```
Worksheets("Sheet1").Columns("A:I").AutoFit
```

This example changes the width of columns A through E on Sheet1 to achieve the best fit, based only on the contents of cells A1:E1.

```
Worksheets("Sheet1").Range("A1:E1").Columns.AutoFit
```

ClearNotes Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearNotesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearNotesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearNotesA "}

Clears notes and sound notes from all the cells in the specified range.

Syntax

expression.**ClearNotes**

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

ClearNotes Method Example

This example clears all notes and sound notes from columns A through C on Sheet1.

```
Worksheets("Sheet1").Columns("A:C").ClearNotes
```

Column Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColumnC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColumnX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColumnA "}

Returns the number of the first column in the first area in the specified range. Read-only **Long**.

Remarks

Column A returns 1, column B returns 2, and so on.

To return the number of the last column in the range, use the following expression.

```
myRange.Columns(myRange.Columns.Count).Column
```

Column Property Example

This example sets the column width of every other column on Sheet1 to 4 points.

```
For Each col In Worksheets("Sheet1").Columns
   If col.Column Mod 2 = 0 Then
      col.ColumnWidth = 4
   End If
Next col
```

ColumnDifferences Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthColumnDifferencesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthColumnDifferencesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthColumnDifferencesA "}

Returns a **Range** object that represents all the cells whose contents are different from the comparison cell in each column.

Syntax

expression.**ColumnDifferences(*Comparison*)**

expression Required. An expression that returns a **Range** object containing the cells to compare.

Comparison Required **Variant**. A single cell to compare to the specified range.

ColumnDifferences Method Example

This example selects the cells in column A on Sheet1 whose contents are different from cell A4.

```
Worksheets("Sheet1").Activate
Set r1 = ActiveSheet.Columns("A").ColumnDifferences( _
   Comparison:=ActiveSheet.Range("A4"))
r1.Select
```

ColumnWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColumnWidthC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColumnWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColumnWidthA "}

Returns or sets the width of all columns in the specified range. Read/write **Variant**.

Remarks

One unit of column width is equal to the width of one character in the Normal style. For proportional fonts, the width of the character 0 (zero) is used.

Use the **Width** property to return the width of a column in points.

If all columns in the range have the same width, the **ColumnWidth** property returns the width. If columns in the range have different widths, this property returns **Null**.

ColumnWidth Property Example

This example doubles the width of column A on Sheet1.

```
With Worksheets("Sheet1").Columns("A")
    .ColumnWidth = .ColumnWidth * 2
End With
```

Consolidate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthConsolidateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthConsolidateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthConsolidateA "}

Consolidates data from multiple ranges on multiple worksheets into a single range on a single worksheet.

Syntax

expression.**Consolidate(*Sources, Function, TopRow, LeftColumn, CreateLinks*)**

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

Sources Optional **Variant**. The sources of the consolidation as an array of text reference strings in R1C1-style notation. The references must include the full path of sheets to be consolidated.

Function Optional **Variant**. The consolidation function. Can be one of the following **XlConsolidationFunction** constants: **xlAverage**, **xlCount**, **xlCountNums**, **xlMax**, **xlMin**, **xlProduct**, **xlStDev**, **xlStDevP**, **xlSum**, **xlVar**, or **xlVarP**. The default value is **xlAverage**.

TopRow Optional **Variant**. **True** to consolidate data based on column titles in the top row of the consolidation ranges. **False** to consolidate data by position. The default value is **False**.

LeftColumn Optional **Variant**. **True** to consolidate data based on row titles in the left column of the consolidation ranges. **False** to consolidate data by position. The default value is **False**.

CreateLinks Optional **Variant**. **True** to have the consolidation use worksheet links. **False** to have the consolidation copy the data. The default value is **False**.

Consolidate Method Example

This example consolidates data from Sheet2 and Sheet3 onto Sheet1, using the SUM function.

```
Worksheets("Sheet1").Range("A1").Consolidate _
    Sources:=Array("Sheet2!R1C1:R37C6", "Sheet3!R1C1:R37C6"), _
    Function:=xlSum
```

CreateNames Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCreateNamesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCreateNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCreateNamesA "}

Creates names in the specified range, based on text labels in the sheet.

Syntax

expression.**CreateNames(*Top, Left, Bottom, Right*)**

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

Top Optional **Variant**. **True** to create names by using labels in the top row. The default value is **False**.

Left Optional **Variant**. **True** to create names by using labels in the left column. The default value is **False**.

Bottom Optional **Variant**.**True** to create names by using labels in the bottom row. The default value is **False**.

Right Optional **Variant**.**True** to create names by using labels in the right column. The default value is **False**.

Remarks

If you don't specify one of ***Top***, ***Left***, ***Bottom***, or ***Right***, Microsoft Excel guesses the location of the text labels, based on the shape of the specified range.

CreateNames Method Example

This example creates names for cells B1:B3 based on the text in cells A1:A3. Note that you must include the cells that contain the names in the range, even though the names are created only for cells B1:B3.

```
Set rangeToName = Worksheets("Sheet1").Range("A1:B3")
rangeToName.CreateNames Left:=True
```

CurrentArray Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCurrentArrayC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCurrentArrayX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCurrentArrayA "}

If the specified cell is part of an array, returns a **Range** object that represents the entire array. Read-only.

CurrentArray Property Example

This example assumes that cell A1 on Sheet1 is the active cell and that the active cell is part of an array that includes cells A1:A10. The example selects cells A1:A10 on Sheet1.

```
ActiveCell.CurrentArray.Select
```

CurrentRegion Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCurrentRegionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCurrentRegionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCurrentRegionA "}

Returns a **Range** object that represents the current region. The current region is a range bounded by any combination of blank rows and blank columns. Read-only.

Remarks

This property is useful for many operations that automatically expand the selection to include the entire current region, such as the **AutoFormat** method.

CurrentRegion Property Example

This example selects the current region on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveCell.CurrentRegion.Select
```

This example assumes that you have a table on Sheet1 that has a header row. The example selects the table, without selecting the header row. The active cell must be somewhere in the table before you run the example.

```
Set tbl = ActiveCell.CurrentRegion
tbl.Offset(1, 0).Resize(tbl.Rows.Count - 1, tbl.Columns.Count).Select
```

DataSeries Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDataSeriesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDataSeriesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDataSeriesA "}

Creates a data series in the specified range.

Syntax

expression.**DataSeries(*Rowcol, Type, Date, Step, Stop, Trend*)**

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

Rowcol Optional **Variant**. Can be the **xlRows** or **xlColumns** constant to have the data series entered in rows or columns, respectively. If this argument is omitted, the size and shape of the range is used.

Type Optional **Variant**. Can be one of the following **XlDataSeriesType** constants: **xlDataSeriesLinear**, **xlGrowth**, **xlChronological**, or **xlAutoFill**. The default value is **xlDataSeriesLinear**.

Date Optional **Variant**. If the ***Type*** argument is **xlChronological**, the ***Date*** argument indicates the step date unit. Can be one of the following **XlDataSeriesDate** constants: **xlDay**, **xlWeekday**, **xlMonth**, or **xlYear**. The default value is **xlDay**.

Step Optional **Variant**. The step value for the series. The default value is 1.

Stop Optional **Variant**. The stop value for the series. If this argument is omitted, Microsoft Excel fills to the end of the range.

Trend Optional **Variant**. **True** to create a linear trend or growth trend. **False** to create a standard data series. The default value is **False**.

DataSeries Method Example

This example creates a series of 12 dates. The series contains the last day of every month in 1996 and is created in the range A1:A12 on Sheet1.

```
Set dateRange = Worksheets("Sheet1").Range("A1:A12")
Worksheets("Sheet1").Range("A1").Formula = "31-JAN-1996"
dateRange.DataSeries Type:=xlChronological, Date:=xlMonth
```

Dependents Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDependentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDependentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDependentsA "}

Returns a **Range** object that represents the range containing all the dependents of a cell. This can be a multiple selection (a union of **Range** objects) if there's more than one dependent. Read-only.

Dependents Property Example

This example selects the dependents of cell A1 on Sheet1.

```
Worksheets("Sheet1").Activate
Range("A1").Dependents.Select
```

DialogBox Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDialogBoxC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDialogBoxX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDialogBoxA "}

Displays a dialog box defined by a dialog box definition table on a Microsoft Excel 4.0 macro sheet. Returns the number of the chosen control, or returns **False** if the user clicks the **Cancel** button.

Syntax

expression.**DialogBox**

expression Required. An expression that returns a **Range** object. The **Range** must refer to a dialog box definition table on a Microsoft Excel 4.0 macro sheet.

DialogBox Method Example

This example runs a Microsoft Excel 4.0 dialog box and then displays the return value in a message box. The `dialogRange` variable refers to the dialog box definition table on the Microsoft Excel 4.0 macro sheet named "Macro1."

```
Set dialogRange = Excel4MacroSheets("Macro1").Range("myDialogBox")
result = dialogRange.DialogBox
MsgBox result
```

DirectDependents Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDirectDependentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDirectDependentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDirectDependentsA "}

Returns a **Range** object that represents the range containing all the direct dependents of a cell. This can be a multiple selection (a union of **Range** objects) if there's more than one dependent. Read-only.

DirectDependents Property Example

This example selects the direct dependents of cell A1 on Sheet1.

```
Worksheets("Sheet1").Activate
Range("A1").DirectDependents.Select
```

DirectPrecedents Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDirectPrecedentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDirectPrecedentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDirectPrecedentsA "}

Returns a **Range** object that represents the range containing all the direct precedents of a cell. This can be a multiple selection (a union of **Range** objects) if there's more than one precedent. Read-only.

DirectPrecedents Property Example

This example selects the direct precedents of cell A1 on Sheet1.

```
Worksheets("Sheet1").Activate
Range("A1").DirectPrecedents.Select
```

EntireColumn Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEntireColumnC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEntireColumnX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEntireColumnA "}

Returns a **Range** object that represents the entire column (or columns) that contains the specified range. Read-only.

EntireColumn Property Example

This example sets the value of the first cell in the column that contains the active cell. The example must be run from a worksheet.

```
ActiveCell.EntireColumn.Cells(1, 1).Value = 5
```

EntireRow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEntireRowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEntireRowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEntireRowA "}

Returns a **Range** object that represents the entire row (or rows) that contains the specified range. Read-only.

EntireRow Property Example

This example sets the value of the first cell in the row that contains the active cell. The example must be run from a worksheet.

```
ActiveCell.EntireRow.Cells(1, 1).Value = 5
```

FillDown Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFillDownC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFillDownX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFillDownA "}

Fills down from the top cell or cells in the specified range to the bottom of the range. The contents and formatting of the cell or cells in the top row of a range are copied into the rest of the rows in the range.

Syntax

expression.**FillDown**

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

FillDown Method Example

This example fills the range A1:A10 on Sheet1, based on the contents of cell A1.

```
Worksheets("Sheet1").Range("A1:A10").FillDown
```

FillLeft Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFillLeftC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFillLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFillLeftA "}

Fills left from the rightmost cell or cells in the specified range. The contents and formatting of the cell or cells in the rightmost column of a range are copied into the rest of the columns in the range.

Syntax

expression.**FillLeft**

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

FillLeft Method Example

This example fills the range A1:M1 on Sheet1, based on the contents of cell M1.

```
Worksheets("Sheet1").Range("A1:M1").FillLeft
```

FillRight Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFillRightC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFillRightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFillRightA "}

Fills right from the leftmost cell or cells in the specified range. The contents and formatting of the cell or cells in the leftmost column of a range are copied into the rest of the columns in the range.

Syntax

expression.**FillRight**

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

FillRight Method Example

This example fills the range A1:M1 on Sheet1, based on the contents of cell A1.

```
Worksheets("Sheet1").Range("A1:M1").FillRight
```

FillUp Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFillUpC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFillUpX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFillUpA "}

Fills up from the bottom cell or cells in the specified range to the top of the range. The contents and formatting of the cell or cells in the bottom row of a range are copied into the rest of the rows in the range.

Syntax

expression.**FillUp**

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

FillUp Method Example

This example fills the range A1:A10 on Sheet1, based on the contents of cell A10.

```
Worksheets("Sheet1").Range("A1:A10").FillUp
```

Find Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFindC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFindX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFindA "}

Finds specific information in a range, and returns a **Range** object that represents the first cell where that information is found. Returns **Nothing** if no match is found. Doesn't affect the selection or the active cell.

For information about using the **Find** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Syntax

expression.**Find(*What, After, LookIn, LookAt, SearchOrder, SearchDirection, MatchCase, MatchByte*)**

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

What Required **Variant**. The data to search for. Can be a string or any Microsoft Excel data type.

After Optional **Variant**. The cell after which you want to search. This corresponds to the position of the active cell when a search is done from the user interface. Note that ***After*** must be a single cell in the range. Remember that the search begins *after* this cell; the specified cell isn't searched until the method wraps back around to this cell. If this argument isn't specified, the search starts after the cell in the upper-left corner of the range.

LookIn Optional **Variant**. Can be one of the following **XlFindLookIn** constants: **xlFormulas**, **xlValues**, or **xlNotes**.

LookAt Optional **Variant**. Can be one of the following **XlLookAt** constants: **xlWhole** or **xlPart**.

SearchOrder Optional **Variant**. Can be one of the following **XlSearchOrder** constants: **xlByRows** or **xlByColumns**.

SearchDirection Optional **Variant**. Can be one of the following **XlSearchDirection** constants: **xlNext** or **xlPrevious**. The default value is **xlNext**.

MatchCase Optional **Variant**. **True** to make the search case sensitive.

MatchByte Optional **Variant**. Used only in the Far East version of Microsoft Excel. **True** to have double-byte characters match only double-byte characters. **False** to have double-byte characters match their single-byte equivalents.

Remarks

The settings for ***LookIn***, ***LookAt***, ***SearchOrder***, ***MatchCase***, and ***MatchByte*** are saved each time you use this method. If you don't specify values for these arguments the next time you call the method, the saved values are used. Setting these arguments changes the settings in the **Find** dialog box, and changing the settings in the **Find** dialog box changes the saved values that are used if you omit the arguments. To avoid problems, set these arguments explicitly each time you use this method.

The **FindNext** and **FindPrevious** methods can be used to repeat the search.

When the search reaches the end of the specified search range, it wraps around to the beginning of the range. To stop a search when this wraparound occurs, save the address of the first found cell, and then test each successive found-cell address against this saved address.

To find cells that match more complicated patterns, use **For Each...Next** with the **Like** operator. For example, the following code searches for all cells in the range A1:C5 that use a font whose name starts with the letters "Cour". When Microsoft Excel finds a match, it changes the font to Times New Roman.

```
For Each c In [A1:C5]
   If c.Font.Name Like "Cour*" Then
      c.Font.Name = "Times New Roman"
   End If
```

Next

Find Method Example

This example finds all cells in the range A1:A500 that contain the value 2 and makes those cells gray.

```
With Worksheets(1).Range("a1:a500")
    Set c = .Find(2, lookin:=xlValues)
    If Not c Is Nothing Then
        firstAddress = c.Address
        Do
            c.Interior.Pattern = xlPatternGray50
            Set c = .FindNext(c)
        Loop While Not c Is Nothing And c.Address <> firstAddress
    End If
End With
```

FindNext Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFindNextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFindNextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFindNextA "}

Continues a search that was begun with the **Find** method. Finds the next cell that matches those same conditions and returns a **Range** object that represents that cell. Doesn't affect the selection or the active cell.

Syntax

expression.**FindNext(*After*)**

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

After Optional **Variant**. The cell after which you want to search. This corresponds to the position of the active cell when a search is done from the user interface. Note that ***After*** must be a single cell in the range. Remember that the search begins *after* this cell; the specified cell isn't searched until the method wraps back around to this cell. If this argument isn't specified, the search starts after the cell in the upper-left corner of the range.

Remarks

When the search reaches the end of the specified search range, it wraps around to the beginning of the range. To stop a search when this wraparound occurs, save the address of the first found cell, and then test each successive found-cell address against this saved address.

FindNext Method Example

This example finds all cells in the range A1:A500 that contain the value 2 and makes those cells gray.

```
With Worksheets(1).Range("a1:a500")
    Set c = .Find(2, lookin:=xlValues)
    If Not c Is Nothing Then
        firstAddress = c.Address
        Do
            c.Interior.Pattern = xlPatternGray50
            Set c = .FindNext(c)
        Loop While Not c Is Nothing And c.Address <> firstAddress
    End If
End With
```

FindPrevious Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFindPreviousC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFindPreviousX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFindPreviousA "}

Continues a search that was begun with the **Find** method. Finds the previous cell that matches those same conditions and returns a **Range** object that represents that cell. Doesn't affect the selection or the active cell.

Syntax

expression.**FindPrevious(*After*)**

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

After Optional **Variant**. The cell before which you want to search. This corresponds to the position of the active cell when a search is done from the user interface. Note that ***After*** must be a single cell in the range. Remember that the search begins *before* this cell; the specified cell isn't searched until the method wraps back around to this cell. If this argument isn't specified, the search starts before the upper- left cell in the range.

Remarks

When the search reaches the beginning of the specified search range, it wraps around to the end of the range. To stop a search when this wraparound occurs, save the address of the first found cell, and then test each successive found-cell address against this saved address.

FindPrevious Method Example

This example shows how the **FindPrevious** method is used with the **Find** and **FindNext** methods. Before running this example, make sure that Sheet1 contains at least two occurrences of the word "Phoenix" in column B.

```
Set fc = Worksheets("Sheet1").Columns("B").Find(what:="Phoenix")
   MsgBox "The first occurrence is in cell " & fc.Address
Set fc = Worksheets("Sheet1").Columns("B").FindNext(after:=fc)
   MsgBox "The next occurrence is in cell " & fc.Address
Set fc = Worksheets("Sheet1").Columns("B").FindPrevious(after:=fc)
   MsgBox "The previous occurrence is in cell " & fc.Address
```

FormulaArray Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormulaArrayC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormulaArrayX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormulaArrayA "}

Returns or sets the array formula of a range. Returns (or can be set to) a single formula or a Visual Basic array. If the specified range doesn't contain an array formula, this property returns **Null**. Read/write **Variant**.

Remarks

If you use this property to enter an array formula, the formula must use the R1C1 reference style, not the A1 reference style (see the second example).

FormulaArray Property Example

This example enters the number 3 as an array constant in cells A1:C5 on Sheet1.

```
Worksheets("Sheet1").Range("A1:C5").FormulaArray = "=3"
```

This example enters the array formula =SUM(R1C1:R3C3) in cells E1:E3 on Sheet1.

```
Worksheets("Sheet1").Range("E1:E3").FormulaArray = _
    "=Sum(R1C1:R3C3)"
```

FormulaHidden Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormulaHiddenC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormulaHiddenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormulaHiddenA "}

True if the formula will be hidden when the workbook or worksheet is protected. Read/write **Boolean**.

Remarks

Don't confuse this property with the **Hidden** property.

FormulaHidden Property Example

This example hides the formulas in column A on Sheet1 when the worksheet is protected.

```
Worksheets("Sheet1").Columns("A").FormulaHidden = True
```

GoalSeek Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGoalSeekC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGoalSeekX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGoalSeekA "}

Calculates the values necessary to achieve a specific goal. If the goal is an amount returned by a formula, this calculates a value that, when supplied to your formula, causes the formula to return the number you want. Returns **True** if the goal seek is successful.

Syntax

expression.**GoalSeek(*Goal, ChangingCell*)**

expression Required. An expression that returns a **Range** object. Must be a single cell.

Goal Required **Variant**. The value you want returned in this cell.

ChangingCell Required **Range**. Specifies which cell should be changed to achieve the target value.

GoalSeek Method Example

This example assumes that Sheet1 has a cell named "Polynomial" that contains the formula =(X^3)+(3*X^2)+6 and another cell named "X" that's empty. The example finds a value for X so that Polynomial contains the value 15.

```
Worksheets("Sheet1").Range("Polynomial").GoalSeek _
   Goal:=15, _
   ChangingCell:=Worksheets("Sheet1").Range("X")
```

Goto Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGotoC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGotoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGotoA "}

Selects any range or Visual Basic procedure in any workbook, and activates that workbook if it's not already active.

Syntax

expression.**Goto(*Reference, Scroll*)**

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

Reference Optional **Variant**. The destination. Can be a **Range** object, a string that contains a cell reference in R1C1-style notation, or a string that contains a Visual Basic procedure name. If this argument is omitted, the destination is the last range you used the **Goto** method to select.

Scroll Optional **Variant**. **True** to scroll through the window so that the upper-left corner of the range appears in the upper-left corner of the window. **False** to not scroll through the window. The default is **False**.

Remarks

This method differs from the **Select** method in the following ways:

- If you specify a range on a sheet that's not on top, Microsoft Excel will switch to that sheet before selecting. (If you use **Select** with a range on a sheet that's not on top, the range will be selected but the sheet won't be activated).
- This method has a ***Scroll*** argument that lets you scroll through the destination window.
- When you use the **Goto** method, the previous selection (before the **Goto** method runs) is added to the array of previous selections (for more information, see the **PreviousSelections** property). You can use this feature to quickly jump between as many as four selections.
- The **Select** method has a ***Replace*** argument; the **Goto** method doesn't.

Goto Method Example

This example selects cell A154 on Sheet1 and then scrolls through the worksheet to display the range.

```
Application.Goto Reference:=Worksheets("Sheet1").Range("A154"), _
   scroll:=True
```

HasArray Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasArrayC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasArrayX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasArrayA "}

True if the specified cell is part of an array formula. Read-only **Variant**.

HasArray Property Example

This example displays a message if the active cell on Sheet1 is part of an array.

```
Worksheets("Sheet1").Activate
If ActiveCell.HasArray =True Then
   MsgBox "The active cell is part of an array"
End If
```

HasFormula Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasFormulaC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasFormulaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasFormulaA "}

True if all cells in the range contain formulas; **False** if none of the cells in the range contains a formula; **Null** otherwise. Read-only **Variant**.

HasFormula Property Example

This example prompts the user to select a range on Sheet1. If every cell in the selected range contains a formula, the example displays a message.

```
Worksheets("Sheet1").Activate
Set rr = Application.InputBox( _
   prompt:="Select a range on this worksheet", _
   Type:=8)
If rr.HasFormula = True Then
   MsgBox "Every cell in the selection contains a formula"
End If
```

Hidden Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHiddenC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHiddenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHiddenA "}

Range object: **True** if the rows or columns are hidden. The specified range must span an entire column or row. Read/write **Variant**.

Scenario object: **True** if the scenario is hidden. The default value is **False**. Read/write **Boolean**.

Remarks

Don't confuse this property with the **FormulaHidden** property.

Hidden Property Example

This example hides column C on Sheet1.

```
Worksheets("Sheet1").Columns("C").Hidden = True
```

Justify Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthJustifyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthJustifyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthJustifyA "}

Rearranges the text in a range so that it fills the range evenly.

Syntax

expression.**Justify**

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

Remarks

If the range isn't large enough, Microsoft Excel displays a message telling you that text will extend below the range. If you click the **OK** button, justified text will replace the contents in cells that extend beyond the selected range. To prevent this message from appearing, set the **DisplayAlerts** property to **False**. After you set this property, text will always replace the contents in cells below the range.

Justify Method Example

This example justifies the text in cell A1 on Sheet1.

```
Worksheets("Sheet1").Range("A1").Justify
```

ListNames Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthListNamesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthListNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthListNamesA "}

Pastes a list of all nonhidden names onto the worksheet, beginning with the first cell in the range.

Syntax

expression.**ListNames**

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

Remarks

Use the **Names** property to return a collection of all the names on a worksheet.

ListNames Method Example

This example pastes a list of defined names into cell A1 on Sheet1. The example pastes both workbook-level names and sheet-level names defined on Sheet1.

```
Worksheets("Sheet1").Range("A1").ListNames
```

Next Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNextA "}

Returns a **Chart**, **Range**, or **Worksheet** object that represents the next sheet or cell. Read-only.

Remarks

If the object is a range, this property emulates the TAB key, although the property returns the next cell without selecting it.

On a protected sheet, this property returns the next unlocked cell. On an unprotected sheet, this property always returns the cell immediately to the right of the specified cell.

Next Property Example

This example selects the next unlocked cell on Sheet1. If Sheet1 is unprotected, this is the cell immediately to the right of the active cell.

```
Worksheets("Sheet1").Activate
ActiveCell.Next.Select
```

ChangeScenario Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthChangeScenarioC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthChangeScenarioX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthChangeScenarioA "}

Changes the scenario to have a new set of changing cells and (optionally) scenario values.

Syntax

expression.**ChangeScenario(*ChangingCells, Values*)**

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

ChangingCells Required **Variant**. A **Range** object that specifies the new set of changing cells for the scenario. The changing cells must be on the same sheet as the scenario.

Values Optional **Variant**. An array that contains the new scenario values for the changing cells. If this argument is omitted, the scenario values are assumed to be the current values in the changing cells.

Remarks

If you specify ***Values***, the array must contain an element for each cell in the ***ChangingCells*** range; otherwise, Microsoft Excel generates an error.

ChangeScenario Method Example

This example sets the changing cells for scenario one to the range A1:A10 on Sheet1.

```
Worksheets("Sheet1").Scenarios(1).ChangeScenario _
    Worksheets("Sheet1").Range("A1:A10")
```

ChangingCells Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChangingCellsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChangingCellsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChangingCellsA "}

Returns a **Range** object that represents the changing cells for a scenario. Read-only.

ChangingCells Property Example

This example selects the changing cells for scenario one on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveSheet.Scenarios(1).ChangingCells.Select
```

CopyPicture Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCopyPictureC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCopyPictureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCopyPictureA "}

Copies the selected object to the Clipboard as a picture.

Syntax

expression.**CopyPicture(*Appearance, Format, Size*)**

expression Required. An expression that returns an object in the Applies To list.

Appearance Optional **Variant**. Specifies how the picture should be copied. Can be one of the following **XlPictureAppearance** constants: **xlScreen** or **xlPrinter**. If **xlScreen** is used, the picture is copied to resemble its display on the screen as closely as possible. If **xlPrinter** is used, the picture is copied as it will look when it's printed. The default value is **xlScreen**.

Format Optional **Variant**. The format of the picture. Can be one of the following **XlCopyPictureFormat** constants: **xlPicture** or **xlBitmap**. The default value is **xlPicture**.

Size Optional **Variant**. Used only with **Chart** objects. The size of the copied picture when the object is a chart on a chart sheet (not embedded on a worksheet). Can be one of the following **XlPictureAppearance** constants: **xlScreen** or **xlPrinter**. If **xlScreen** is used, the picture is copied to match the size of its display on the screen as closely as possible. If **xlPrinter** is used, the picture is copied to match its printed size as closely as possible. The default value is **xlPrinter**.

Remarks

If you copy a range, it must be made up of adjacent cells.

CopyPicture Method Example

This example copies a screen image of cells A1:D4 on Sheet1 to the Clipboard, and then it pastes the bitmap to another location on Sheet1.

```
Worksheets("Sheet1").Range("A1:D4").CopyPicture xlScreen, xlBitmap
Worksheets("Sheet1").Paste _
   Destination:=Worksheets("Sheet1").Range("E6")
```

CreateSummary Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCreateSummaryC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCreateSummaryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCreateSummaryA "}

Creates a new worksheet that contains a summary report for the scenarios on the specified worksheet.

Syntax

expression.**CreateSummary**(***ReportType, ResultCells***)

expression Required. An expression that returns a **Scenarios** collection.

ReportType Optional **Variant**. The report type. Can be one of the following **XlSummaryReportType** constants: **xlStandardSummary** or **xlSummaryPivotTable**.The default value is **xlStandardSummary**.

ResultCells Optional **Variant**. A **Range** object that represents the result cells on the specified worksheet. Normally, this range refers to one or more cells containing the formulas that depend on the changing cell values for your model – that is, the cells that show the results of a particular scenario. If this argument is omitted, there are no result cells included in the report.

CreateSummary Method Example

This example creates a summary of the scenarios on Sheet1, with result cells in the range C4:C9 on Sheet1.

```
Worksheets("Sheet1").Scenarios.CreateSummary _
   ResultCells := Worksheets("Sheet1").Range("C4:C9")
```

Cut Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCutC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCutA "}

Cuts the object to the Clipboard or pastes it into a specified destination.

Syntax

expression.**Cut(*Destination*)**

expression Required. An expression that returns an object in the Applies To list.

Destination Optional **Variant**. Used only with **Range** objects. The range where the object should be pasted. If this argument is omitted, the object is cut to the Clipboard.

Remarks

The cut range must be made up of adjacent cells.

Only embedded charts can be cut.

Cut Method Example

This example cuts the range A1:G37 on Sheet1 and places it on the Clipboard.

```
Worksheets("Sheet1").Range("A1:G37").Cut
```

Delivery Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDeliveryC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDeliveryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDeliveryA "}

Returns or sets the routing delivery method. Can be one of the following **XlRoutingSlipDelivery** constants: **xlOneAfterAnother** or **xlAllAtOnce**. Read/write **Long**.

Remarks

You cannot set this property if routing is in progress

Delivery Property Example

This example sends Book1.xls to three recipients, one after another.

```
Workbooks("BOOK1.XLS").HasRoutingSlip = True
With Workbooks("BOOK1.XLS").RoutingSlip
    .Delivery = xlOneAfterAnother
    .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
    .Subject = "Here is BOOK1.XLS"
    .Message = "Here is the workbook. What do you think?"
End With
Workbooks("BOOK1.XLS").Route
```

ErrorBar Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthErrorBarC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthErrorBarX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthErrorBarA "}

Applies error bars to the series.

Syntax

expression.**ErrorBar(*Direction, Include, Type, Amount, MinusValues*)**

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

Direction Optional **Variant**. The error bar direction. Can be one of the following **XlErrorBarDirection** constants: **xlX** or **xlY**. **xlX** can only be used with scatter charts.The default value is **xlY**.

Include Optional **Variant**. The error bar parts to include. Can be one of the following **XlErrorBarInclude** constants: **xlErrorBarIncludePlusValues**, **xlErrorBarIncludeMinusValues**, **xlErrorBarIncludeNone**, or **xlErrorBarIncludeBoth**. The default value is **xlErrorBarIncludeBoth**.

Type Optional **Variant**. The error bar type. Can be one of the following **XlErrorBarType** constants: **xlErrorBarTypeFixedValue**, **xlErrorBarTypePercent**, **xlErrorBarTypeStDev**, **xlErrorBarTypeStError**, or **xlErrorBarTypeCustom**.

Amount Optional **Variant**. The error amount. Used for only the positive error amount when ***Type*** is **xlErrorBarTypeCustom**.

MinusValues Optional **Variant**. The negative error amount when ***Type*** is **xlErrorBarTypeCustom**.

ErrorBar Method Example

This example applies standard error bars in the Y direction for series one in Chart1. The error bars are applied in the positive and negative directions. The example should be run on a 2-D line chart.

```
Charts("Chart1").SeriesCollection(1).ErrorBar _
   Direction:=xlY, Include:=xlErrorBarIncludeBoth, _
   Type:=xlErrorBarTypeStError
```

ErrorBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproErrorBarsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproErrorBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproErrorBarsA "}

Returns an **ErrorBars** object that represents the error bars for the series. Read-only.

ErrorBars Property Example

This example sets the error bar color for series one in Chart1. The example should be run on a 2-D line chart that has error bars for series one.

```
With Charts("Chart1").SeriesCollection(1)
   .ErrorBars.Border.ColorIndex = 8
End With
```

Formula Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormulaC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormulaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormulaA "}

Returns or sets the object's formula, in A1-style notation and in the language of the macro. Read/write **Variant** for **Range** objects, read/write **String** for all other objects.

Remarks

If the cell contains a constant, this property returns the constant. If the cell is empty, **Formula** returns an empty string. If the cell contains a formula, **Formula** returns the formula as a string, in the same format in which it would be displayed in the formula bar (including the equal sign).

If you set the value or formula of a cell to a date, Microsoft Excel checks to see whether that cell is already formatted with one of the date or time number formats. If not, Microsoft Excel changes the number format to the default short date number format.

If the range is a one- or two-dimensional range, you can set the formula to a Visual Basic array of the same dimensions. Similarly, you can put the formula into a Visual Basic array.

Setting the formula for a multiple-cell range fills all cells in the range with the formula.

Formula Property Example

This example sets the formula for cell A1 on Sheet1.

```
Worksheets("Sheet1").Range("A1").Formula = "=$A$4+$A$10"
```

FunctionWizard Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFunctionWizardC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFunctionWizardX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFunctionWizardA "}

Starts the Function Wizard for the upper-left cell of the range.

Syntax

expression.**FunctionWizard**

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

FunctionWizard Method Example

This example starts the Function Wizard for the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveCell.FunctionWizard
```

HasErrorBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasErrorBarsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasErrorBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasErrorBarsA "}

True if the series has error bars. This property isn't available for 3-D charts. Read/write **Boolean**.

HasErrorBars Property Example

This example removes error bars from series one in Chart1. The example should be run on a 2-D line chart that has error bars for series one.

```
Charts("Chart1").SeriesCollection(1).HasErrorBars = False
```

HorizontalAlignment Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHorizontalAlignmentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHorizontalAlignmentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHorizontalAlignmentA "}

Returns or sets the horizontal alignment for the object. For all objects, this can be one of the following **XlHAlign** constants: **xlHAlignCenter**, **xlHAlignDistributed**, **xlHAlignJustify**, **xlHAlignLeft**, or **xlHAlignRight**. In addition, for the **Range** or **Style** object, this property can be set to **xlHAlignCenterAcrossSelection**, **xlHAlignFill**, or **xlHAlignGeneral**. Read/write **Long**.

Remarks

The **xlHAlignDistributed** alignment style works only in Far East versions of Microsoft Excel.

HorizontalAlignment Property Example

This example left aligns the range A1:A5 on Sheet1.

```
Worksheets("Sheet1").Range("A1:A5").HorizontalAlignment = xlLeft
```

Import Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthImportC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthImportX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthImportA "}

This method should not be used. Sound notes have been removed from Microsoft Excel.

Interior Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInteriorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInteriorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInteriorA "}

Returns an **Interior** object that represents the interior of the specified object. Read-only.

Interior Property Example

This example sets the interior color for cell A1 on Sheet1 to cyan.

```
Worksheets("Sheet1").Range("A1").Interior.ColorIndex = 8
```

InvertIfNegative Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInvertIfNegativeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInvertIfNegativeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInvertIfNegativeA "}

True if Microsoft Excel inverts the pattern in the item when it corresponds to a negative number. Read/write **Variant** for the **Interior** object, read/write **Boolean** for all other objects.

InvertIfNegative Property Example

This example inverts the pattern for negative values in series one in Chart1. The example should be run on a 2-D column chart.

```
Charts("Chart1").SeriesCollection(1).InvertIfNegative = True
```

Locked Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLockedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLockedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLockedA "}

True if the object is locked, **False** if the object can be modified when the sheet is protected. Returns **Null** if the specified range contains both locked and unlocked cells. Read/write **Variant** for the **Range** object; read/write **Boolean** for all other objects.

Locked Property Example

This example unlocks cells A1:G37 on Sheet1 so that they can be modified when the sheet is protected.

```
Worksheets("Sheet1").Range("A1:G37").Locked = False
Worksheets("Sheet1").Protect
```

Message Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMessageC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMessageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMessageA "}

Returns or sets the message text for the routing slip. This text is used as the body text of mail messages that are used to route the workbook. Read/write **String**.

Message Property Example

This example sends Book1.xls to three recipients, one after another.

```
Workbooks("BOOK1.XLS").HasRoutingSlip = True
With Workbooks("BOOK1.XLS").RoutingSlip
    .Delivery = xlOneAfterAnother
    .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
    .Subject = "Here is BOOK1.XLS"
    .Message = "Here is the workbook. What do you think?"
End With
Workbooks("BOOK1.XLS").Route
```

NumberFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNumberFormatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNumberFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNumberFormatA "}

Returns or sets the format code for the object. Returns **Null** if all cells in the specified range don't have the same number format. Read/write **Variant** for the **Range** object, read/write **String** for all other objects.

Remarks

For the **PivotField** object, you can set the **NumberFormat** property only for a data field.

The format code is the same string as the **Format Codes** option in the **Format Cells** dialog box. The **Format** function uses different format code strings than do the **NumberFormat** and **NumberFormatLocal** properties.

NumberFormat Property Example

These examples set the number format for cell A17, row one, and column C (respectively) on Sheet1.

```
Worksheets("Sheet1").Range("A17").NumberFormat = "General"
Worksheets("Sheet1").Rows(1).NumberFormat = "hh:mm:ss"
Worksheets("Sheet1").Columns("C"). _
    NumberFormat = "$#,##0.00_);[Red]($#,##0.00)"
```

Play Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPlayC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPlayX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPlayA "}

This method should not be used. Sound notes have been removed from Microsoft Excel.

PlotOrder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPlotOrderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPlotOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPlotOrderA "}

Returns or sets the plot order for the selected series within the chart group. Read/write **Long**.

Remarks

You can set plot order only within a chart group (you cannot set the plot order for the entire chart if you have more than one chart type). A chart group is a collection of series with the same chart type.

Changing the plot order of one series will cause the plot orders of the other series in the chart group to be adjusted, as necessary.

PlotOrder Property Example

This example makes series two in Chart1 appear third in the plot order. The example should be run on a 2-D column chart that contains three or more series.

```
Charts("Chart1").ChartGroups(1).SeriesCollection(2).PlotOrder = 3
```

Points Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPointsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPointsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPointsA "}

Returns an object that represents a single point (a **Point** object, Syntax 1) or a collection of all the points (a **Points** object, Syntax 2) in the series. Read-only.

Syntax 1

expression.**Points(*Index*)**

Syntax 2

expression.**Points**

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

Index Optional **Variant**. The name or number of the point.

Points Method Example

This example applies a data label to point one in series one in Chart1.

```
Charts("Chart1").SeriesCollection(1).Points(1).ApplyDataLabels
```

Previous Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPreviousC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPreviousX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPreviousA "}

Returns a **Chart**, **Range**, or **Worksheet** object that represents the previous sheet or cell. Read-only.

Remarks

If the object is a range, this property emulates pressing SHIFT+TAB; unlike the key combination, however, the property returns the previous cell without selecting it.

On a protected sheet, this property returns the previous unlocked cell. On an unprotected sheet, this property always returns the cell immediately to the left of the specified cell.

Previous Property Example

This example selects the previous unlocked cell on Sheet1. If Sheet1 is unprotected, this is the cell immediately to the left of the active cell.

```
Worksheets("Sheet1").Activate
ActiveCell.Previous.Select
```

Record Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRecordC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRecordX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRecordA "}

This method should not be used. Sound notes have been removed from Microsoft Excel.

RemoveSubtotal Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRemoveSubtotalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRemoveSubtotalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRemoveSubtotalA "}

Removes subtotals from a list.

Syntax

expression.**RemoveSubtotal**

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

RemoveSubtotal Method Example

This example removes subtotals from the range A1:G37 on Sheet1. The example should be run on a list that has subtotals.

```
Worksheets("Sheet1").Range("A1:G37").RemoveSubtotal
```

Replace Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthReplaceC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthReplaceX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthReplaceA "}

Finds and replaces characters in cells within a range. Doesn't change the selection or the active cell.

For information about using the **Replace** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Syntax

expression.**Replace(*What, Replacement, LookAt, SearchOrder, MatchCase, MatchByte*)**

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

What Required **String**. The string to search for.

Replacement Required **String**. The replacement string.

LookAt Optional **Variant**. Can be one of the following **XlLookAt** constants: **xlWhole** or **xlPart**.

SearchOrder Optional **Variant**. Can be one of the following **XlSearchOrder** constants: **xlByRows** or **xlByColumns**.

MatchCase Optional **Variant**. **True** to make the search case sensitive.

MatchByte Optional **Variant**. Used only in Far East versions of Microsoft Excel. **True** to have double-byte characters match only double-byte characters. **False** to have double-byte characters match their single-byte equivalents.

Remarks

The settings for ***LookAt***, ***SearchOrder***, ***MatchCase***, and ***MatchByte*** are saved each time you use this method. If you don't specify values for these arguments the next time you call the method, the saved values are used. Setting these arguments changes the settings in the **Find** dialog box, and changing the settings in the **Find** dialog box changes the saved values that are used if you omit the arguments. To avoid problems, set these arguments explicitly each time you use this method.

If the contents of the ***What*** argument are found in at least one cell on the sheet, this method returns **True**.

Replace Method Example

This example replaces every occurrence of the function SIN with the function COS. The replacement range is column A on Sheet1.

```
Worksheets("Sheet1").Columns("A").Replace _
   What:="SIN", Replacement:="COS", _
   SearchOrder:=xlByColumns, MatchCase:=True
```

Reset Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthResetC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthResetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthResetA "}

Resets the routing slip so that a new routing can be initiated with the same slip (using the same recipient list and delivery information). The routing must be completed before you use this method. Using this method at other times causes an error.

Syntax

expression.**Reset**

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

Reset Method Example

This example resets the routing slip for Book1.xls if routing has been completed.

```
With Workbooks("BOOK1.XLS").RoutingSlip
   If .Status = xlRoutingComplete Then
      .Reset
   Else
      MsgBox "Cannot reset routing; not yet complete"
   End If
End With
```

Resize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproResizeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproResizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproResizeA "}

Resizes the specified range. Returns a **Range** object that represents the resized range.

Syntax

expression.**Resize(*RowSize, ColumnSize*)**

expression Required. An expression that returns a **Range** object to be resized.

RowSize Optional **Variant**. The number of rows in the new range. If this argument is omitted, the number of rows in the range remains the same.

ColumnSize Optional **Variant**. The number of columns in the new range. If this argument is omitted, the number of columns in the range remains the same.

Resize Property Example

This example resizes the selection on Sheet1 to extend it by one row and one column.

```
Worksheets("Sheet1").Activate
numRows = Selection.Rows.Count
numColumns = Selection.Columns.Count
Selection.Resize(numRows + 1, numColumns + 1).Select
```

This example assumes that you have a table on Sheet1 that has a header row. The example selects the table, without selecting the header row. The active cell must be somewhere in the table before you run the example.

```
Set tbl = ActiveCell.CurrentRegion
tbl.Offset(1, 0).Resize(tbl.Rows.Count - 1, tbl.Columns.Count).Select
```

ReturnWhenDone Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReturnWhenDoneC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReturnWhenDoneX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReturnWhenDoneA "}

True if the workbook is returned to the sender when routing is finished. Read/write **Boolean**.

Remarks

You cannot set this property if routing is in progress

ReturnWhenDone Property Example

This example sends Book1.xls to three recipients, one after another, and then it returns the workbook to the sender when routing has been completed.

```
Workbooks("BOOK1.XLS").HasRoutingSlip = True
With Workbooks("BOOK1.XLS").RoutingSlip
    .Delivery = xlOneAfterAnother
    .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
    .Subject = "Here is BOOK1.XLS"
    .Message = "Here is the workbook. What do you think?"
    .ReturnWhenDone = True
End With
Workbooks("BOOK1.XLS").Route
```

Row Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowA "}

Returns the number of the first row of the first area in the range. Read-only **Long**.

Row Property Example

This example sets the row height of every other row on Sheet1 to 4 points.

```
For Each rw In Worksheets("Sheet1").Rows
   If rw.Row Mod 2 = 0 Then
      rw.RowHeight = 4
   End If
Next rw
```

RowDifferences Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRowDifferencesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRowDifferencesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRowDifferencesA "}

Returns a **Range** object that represents all the cells whose contents are different from those of the comparison cell in each row.

Syntax

expression.**RowDifferences(*Comparison*)**

expression Required. An expression that returns a range containing the cells to be compared.

Comparison Required **Variant**. A single cell to compare with the specified range.

RowDifferences Method Example

This example selects the cells in row one on Sheet1 whose contents are different from those of cell D1.

```
Worksheets("Sheet1").Activate
Set c1 = ActiveSheet.Rows(1).RowDifferences( _
   comparison:=ActiveSheet.Range("D1"))
c1.Select
```

RowHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowHeightC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowHeightA "}

Returns the height of all the rows in the range specified, measured in points. Returns **Null** if the rows in the specified range aren't all the same height. Read/write **Variant**.

Remarks

For a single row, the value of the **Height** property is equal to the value of the **RowHeight** property. However, you can also use the **Height** property to return the total height of a range of cells.

Other differences between **RowHeight** and **Height** include the following:

- **Height** is read-only.
- If you return the **RowHeight** property of several rows, you will either get the row height of each of the rows (if all the rows are the same height) or **Null** (if they're different heights). If you return the **Height** property of several rows, you will get the total height of all the rows.

RowHeight Property Example

This example doubles the height of row one on Sheet1.

```
With Worksheets("Sheet1").Rows(1)
    .RowHeight = .RowHeight * 2
End With
```

ShowDependents Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowDependentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowDependentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowDependentsA "}

Draws tracer arrows to the direct dependents of the range.

Syntax

expression.**ShowDependents**(***Remove***)

expression Required. An expression that returns a **Range** object. Must be a single cell.

Remove Optional **Variant**. **True** to remove one level of tracer arrows to direct dependents. **False** to expand one level of tracer arrows. The default value is **False**.

ShowDependents Method Example

This example draws tracer arrows to dependents of the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveCell.ShowDependents
```

This example removes the tracer arrow for one level of dependents of the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveCell.ShowDependents Remove:=True
```

ShowErrors Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowErrorsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowErrorsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowErrorsA "}

Draws tracer arrows through the precedents tree to the cell that's the source of the error, and returns the range that contains that cell.

Syntax

expression.**ShowErrors**

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

ShowErrors Method Example

This example displays a red tracer arrow if there's an error in the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
If IsError(ActiveCell.Value) Then
   ActiveCell.ShowErrors
End If
```

ShowPrecedents Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowPrecedentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowPrecedentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowPrecedentsA "}

Draws tracer arrows to the direct precedents of the range.

Syntax

expression.**ShowPrecedents(*Remove*)**

expression Required. An expression that returns a **Range** object. Must be a single cell.

Remove Optional **Variant**. **True** to remove one level of tracer arrows to direct precedents. **False** to expand one level of tracer arrows. The default value is **False**.

ShowPrecedents Method Example

This example draws tracer arrows to the precedents of the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveCell.ShowPrecedents
```

This example removes the tracer arrow for one level of precedents of the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveCell.ShowPrecedents remove:=True
```

Smooth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSmoothC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSmoothX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSmoothA "}

True if curve smoothing is turned on for the line chart or scatter chart. Applies only to line and scatter charts. Read/write.

Smooth Property Example

This example turns on curve smoothing for series one in Chart1. The example should be run on a 2-D line chart.

```
Charts("Chart1").SeriesCollection(1).Smooth = True
```

Sort Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSortC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSortX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSortA "}

Sorts a PivotTable, a range, or the current region (if the specified range contains only one cell).

Syntax

expression.**Sort(*Key1, Order1, Key2, Type, Order2, Key3, Order3, Header, OrderCustom, MatchCase, Orientation, SortMethod, IgnoreControlCharacters, IgnoreDiacritics, IgnoreKashida*)**

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

Key1 Optional **Variant**. The first sort field, as either text (a pivot field or range name) or a **Range** object ("Dept" or `Cells(1, 1)`, for example).

Order1 Optional **Variant**. Can be one of the following **XlSortOrder** constants: **xlAscending** or **xlDescending**. Use **xlAscending** to sort ***Key1*** in ascending order. Use **xlDescending** to sort ***Key1*** in descending order. The default value is **xlAscending**.

Key2 Optional **Variant**. The second sort field, as either text (a pivot field or range name) or a **Range** object. If this argument is omitted, there's no second sort field. Not used when sorting PivotTables.

Type Optional **Variant**. Specifies which elements are sorted. Can be one of the following **XlSortType** constants: **xlSortValues** or **xlSortLabels**. Used only when sorting PivotTables.

Order2 Optional **Variant**. Can be one of the following **XlSortOrder** constants: **xlAscending** or **xlDescending**. Use **xlAscending** to sort ***Key2*** in ascending order. Use **xlDescending** to sort ***Key2*** in descending order. The default value is **xlAscending**. Not used when sorting PivotTables.

Key3 Optional **Variant**. The third sort field, as either text (a range name) or a **Range** object. If this argument is omitted, there's no third sort field. Not used when sorting PivotTables.

Order3 Optional **Variant**. Can be one of the following **XlSortOrder** constants: **xlAscending** or **xlDescending**. Use **xlAscending** to sort ***Key3*** in ascending order. Use **xlDescending** to sort ***Key3*** in descending order. The default value is **xlAscending**. Not used when sorting PivotTables.

Header Optional **Variant**. Specifies whether the first row contains headers. Can be one of the following **XlYesNoGuess** constants: **xlYes**, **xlNo**, or **xlGuess**. Use **xlYes** if the first row contains headers (it shouldn't be sorted). Use **xlNo** if there are no headers (the entire range should be sorted). Use **xlGuess** to let Microsoft Excel determine whether there's a header, and to determine where it is, if there is one. The default value is **xlNo**. Not used when sorting PivotTables.

OrderCustom Optional **Variant**. A 1-based integer offset nto the list of custom sort orders. If this argument is omitted, 1 (Normal) is used.

MatchCase Optional **Variant**. **True** to do a case-sensitive sort; **False** to do a sort that's not case sensitive. Not used when sorting PivotTables.

Orientation Optional **Variant**. If **xlTopToBottom** is used, the sort is done from top to bottom (by row). If **xlLeftToRight** is used, the sort is done from left to right (by column).

SortMethod Optional **Variant**. The type of sort. Can be one of the following **XlSortMethod** constants: **xlSyllabary** (to sort phonetically) or **xlCodePage** (to sort by code page). The default value is **xlSyllabary**.

IgnoreControlCharacters Optional **Variant**. Not used in US/English Microsoft Excel.

IgnoreDiacritics Optional **Variant**. Not used in US/English Microsoft Excel.

IgnoreKashida Optional **Variant**. Not used in US/English Microsoft Excel.

Sort Method Example

This example sorts the range A1:C20 on Sheet1, using cell A1 as the first sort key and cell B1 as the second sort key. The sort is done in ascending order by row, and there are no headers.

```
Worksheets("Sheet1").Range("A1:C20").Sort _
   Key1:=Worksheets("Sheet1").Range("A1"), _
   Key2:=Worksheets("Sheet1").Range("B1")
```

This example sorts the current region that contains cell A1 on Sheet1, sorting by the data in the first column and automatically using a header row if one exists. The **Sort** method determines the current region automatically.

```
Worksheets("Sheet1").Range("A1").Sort _
   Key1:=Worksheets("Sheet1").Columns("A"), _
   Header:=xlGuess
```

SoundNote Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSoundNoteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSoundNoteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSoundNoteA "}

This property should not be used. Sound notes have been removed from Microsoft Excel.

Status Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStatusC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStatusX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStatusA "}

Indicates the status of the routing slip. Can be one of the following **XlRoutingSlipStatus** constants: **xlNotYetRouted**, **xlRoutingInProgress**, or **xlRoutingComplete**. Read-only **Long**.

Status Property Example

This example resets the routing slip for Book1.xls if routing has been completed.

```
With Workbooks("BOOK1.XLS").RoutingSlip
   If .Status = xlRoutingComplete Then
      .Reset
   Else
      MsgBox "Cannot reset routing; not yet complete."
   End If
End With
```

Style Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStyleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStyleA "}

Returns a **Style** object that represents the style of the specified range. Read-only.

Style Property Example

This example applies the Normal style to cell A1 on Sheet1.

```
Worksheets("Sheet1").Range("A1").Style.Name = "Normal"
```

If cell B4 on Sheet1 currently has the Normal style applied, this example applies the Percent style.

```
If Worksheets("Sheet1").Range("B4").Style.Name = "Normal" Then
   Worksheets("Sheet1").Range("B4").Style.Name = "Percent"
End If
```

SubscribeTo Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSubscribeToC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSubscribeToX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSubscribeToA "}

Macintosh only. Subscribes to a published edition.

Syntax

expression.**SubscribeTo(*Edition, Format*)**

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

Edition Required **String**. The name of the edition to which you want to subscribe.

Format Optional **Variant**. Can be one of the following **XlSubscribeToFormat** constants: **xlSubscribeToPicture** or **xlSubscribeToText**.

SubscribeTo Method Example

This example subscribes to the "stock data" publisher.

```
Worksheets("Sheet1").Range("D1").SubscribeTo "stock data"
```

Subtotal Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSubtotalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSubtotalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSubtotalA "}

Creates subtotals for the range (or the current region, if the range is a single cell).

For information about using the **Subtotal** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Syntax

expression.**Subtotal(*GroupBy, Function, TotalList, Replace, PageBreaks, SummaryBelowData*)**

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

GroupBy Required **Long**. The field to group by, as a 1-based integer offset. For more information, see the example.

Function Required **Long**. The subtotal function. Can be one of the following **XlConsolidationFunction** constants: **xlAverage**, **xlCount**, **xlCountNums**, **xlMax**, **xlMin**, **xlProduct**, **xlStDev**, **xlStDevP**, **xlSum**, **xlVar**, or **xlVarP**.

TotalList Required **Variant**. An array of 1-based field offsets, indicating the fields to which the subtotals are added. For more information, see the example.

Replace Optional **Variant**. **True** to replace existing subtotals. The default value is **False**.

PageBreaks Optional **Variant**. **True** to add page breaks after each group. The default value is **False**.

SummaryBelowData Optional **Variant**. Can be one of the following **xlSummaryRow** constants: **xlSummaryAbove** or **xlSummaryBelow**. The default value is **xlSummaryBelow**.

Subtotal Method Example

This example creates subtotals for the selection on Sheet1. The subtotals are sums grouped by each change in field one, with the subtotals added to fields two and three.

```
Worksheets("Sheet1").Activate
Selection.Subtotal groupBy:=1, function:=xlSum, _
   totalList:=Array(2, 3)
```

Summary Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSummaryC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSummaryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSummaryA "}

True if the range is an outlining summary row or column. The range should be a row or a column. Read-only **Variant**.

Summary Property Example

This example formats row four on Sheet1 as bold and italic if it's an outlining summary column.

```
With Worksheets("Sheet1").Rows(4)
   If .Summary = True Then
      .Font.Bold = True
      .Font.Italic = True
   End If
End With
```

Table Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthTableC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthTableA "}

Creates a data table based on input values and formulas that you define on a worksheet.

Syntax

expression.**Table(*RowInput, ColumnInput*)**

expression Required. An expression that returns an object in the Applies To list.

RowInput Optional **Variant**. A single cell to use as the row input for your table.

ColumnInput Optional **Variant**. A single cell to use as the column input for your table.

Remarks

Use data tables to perform a what-if analysis by changing certain constant values on your worksheet to see how values in other cells are affected.

Table Method Example

This example creates a formatted multiplication table in cells A1:K11 on Sheet1.

```
Set dataTableRange = Worksheets("Sheet1").Range("A1:K11")
Set rowInputCell = Worksheets("Sheet1").Range("A12")
Set columnInputCell = Worksheets("Sheet1").Range("A13")

Worksheets("Sheet1").Range("A1").Formula = "=A12*A13"
For i = 2 To 11
   Worksheets("Sheet1").Cells(i, 1) = i - 1
   Worksheets("Sheet1").Cells(1, i) = i - 1
Next i
dataTableRange.Table rowInputCell, columnInputCell
With Worksheets("Sheet1").Range("A1").CurrentRegion
   .Rows(1).Font.Bold = True
   .Columns(1).Font.Bold = True
   .Columns.AutoFit
End With
```

TrackStatus Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTrackStatusC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTrackStatusX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTrackStatusA "}

True if status tracking is enabled for the routing slip. Read/write **Boolean**.

Remarks

You cannot set this property if routing is in progress

TrackStatus Property Example

This example sends Book1.xls to three recipients, with status tracking enabled.

```
Workbooks("BOOK1.XLS").HasRoutingSlip = True
With Workbooks("BOOK1.XLS").RoutingSlip
    .Delivery = xlOneAfterAnother
    .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
    .Subject = "Here is BOOK1.XLS"
    .Message = "Here is the workbook. What do you think?"
    .ReturnWhenDone = True
    .TrackStatus = True
End With
Workbooks("BOOK1.XLS").Route
```

Trendlines Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthTrendlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthTrendlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthTrendlinesA "}

Returns an object that represents a single trendline (a **Trendline** object, Syntax 1) or a collection of all the trendlines (a **Trendlines** object, Syntax 2) for the series.

Syntax 1

object.**Trendlines(*Index*)**

Syntax 2

object.**Trendlines**

object Required. The **Series** object.

Index Optional **Variant**. The name or number of the trendline.

Trendlines Method Example

This example adds a linear trendline to series one in Chart1.

```
Charts("Chart1").SeriesCollection(1).Trendlines.Add Type:=xlLinear
```

UseStandardHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUseStandardHeightC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUseStandardHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUseStandardHeightA "}

True if the row height of the **Range** object equals the standard height of the sheet. Returns **Null** if the range contains more than one row and the rows aren't all the same height. Read/write **Variant**.

UseStandardHeight Property Example

This example sets the height of row one on Sheet1 to the standard height.

```
Worksheets("Sheet1").Rows(1).UseStandardHeight = True
```

UseStandardWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUseStandardWidthC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUseStandardWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUseStandardWidthA "}

True if the column width of the **Range** object equals the standard width of the sheet. Returns **Null** if the range contains more than one column and the columns aren't all the same width. Read/write **Variant**.

UseStandardWidth Property Example

This example sets the width of column A on Sheet1 to the standard width.

```
Worksheets("Sheet1").Columns("A").UseStandardWidth = True
```

Values Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproValuesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproValuesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproValuesA "}

Scenario object: Returns an array that contains the current values of the changing cells for the scenario. Read-only **Variant**.

Series object: Returns or sets a collection of all the values in the series. This can be a range on a worksheet or an array of constant values, but not a combination of both. See the examples for details. Read/write **Variant**.

Values Property Example

This example sets the series values from a range.

```
Charts("Chart1").SeriesCollection(1).Values = _
   Worksheets("Sheet1").Range("C5:T5")
```

To assign a constant value to each individual data point, you must use an array.

```
Charts("Chart1").SeriesCollection(1).Values = _
   Array(1, 3, 5, 7, 11, 13, 17, 19)
```

VerticalAlignment Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVerticalAlignmentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVerticalAlignmentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVerticalAlignmentA "}

Returns or sets the vertical alignment of the object. Can be one of the following **XlVAlign** constants: **xlVAlignBottom**, **xlVAlignCenter**, **xlVAlignDistributed**, **xlVAlignJustify**, or **xlVAlignTop**. Read/write **Long**.

Remarks

The **xlVAlignDistributed** alignment style works only in Far East versions of Microsoft Excel.

VerticalAlignment Property Example

This example sets the height of row two on Sheet1 to twice the standard height and then centers the contents of the row vertically.

```
Worksheets("Sheet1").Rows(2).RowHeight = _
   2 * Worksheets("Sheet1").StandardHeight
Worksheets("Sheet1").Rows(2).VerticalAlignment = xlVAlignCenter
```

Worksheet Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWorksheetC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWorksheetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWorksheetA "}

Returns a **Worksheet** object that represents the worksheet containing the specified range. Read-only.

Worksheet Property Example

This example displays the name of the worksheet that contains the active cell. The example must be run from a worksheet.

```
MsgBox ActiveCell.Worksheet.Name
```

This example displays the name of the worksheet that contains the range named "testRange."

```
MsgBox Range("testRange").Worksheet.Name
```

WrapText Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWrapTextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWrapTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWrapTextA "}

True if Microsoft Excel wraps the text in the object. Returns **Null** if the specified range contains some cells that wrap text and other cells that don't. Read/write **Variant** for the **Range** object, read/write **Boolean** for the **Style** object.

Remarks

Microsoft Excel will change the row height of the range, if necessary, to accommodate the text in the range.

WrapText Property Example

This example formats cell B2 on Sheet1 so that the text wraps within the cell.

```
Worksheets("Sheet1").Range("B2").Value = _
   "This text should wrap in a cell."
Worksheets("Sheet1").Range("B2").WrapText = True
```

Using Microsoft Excel Worksheet Functions in Visual Basic

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscUsingWorksheetFunctionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmscUsingWorksheetFunctionsX":1}

You can use most Microsoft Excel worksheet functions in your Visual Basic statements. To see a list of the worksheet functions you can use, see List of Worksheet Functions Available to Visual Basic.

Note Some worksheet functions aren't useful in Visual Basic. For example, the **Concatenate** function isn't needed because in Visual Basic you can use the **&** operator to join multiple text values.

Calling a Worksheet Function from Visual Basic

In Visual Basic, the Microsoft Excel worksheet functions are available through the **WorksheetFunction** object.

The following **Sub** procedure uses the **Min** worksheet function to determine the smallest value in a range of cells. First, the variable `myRange` is declared as a **Range** object, and then it's set to range A1:C10 on Sheet1. Another variable, `answer`, is assigned the result of applying the **Min** function to `myRange`. Finally, the value of `answer` is displayed in a message box.

```
Sub UseFunction()
   Dim myRange As Range
   Set myRange = Worksheets("Sheet1").Range("A1:C10")
   answer = Application.WorksheetFunction.Min(myRange)
   MsgBox answer
End Sub
```

If you use a worksheet function that requires a range reference as an argument, you must specify a **Range** object. For example, you can use the **Match** worksheet function to search a range of cells. In a worksheet cell, you would enter a formula such as =MATCH(9,A1:A10,0). However, in a Visual Basic procedure, you would specify a **Range** object to get the same result.

```
Sub FindFirst()
   myVar = Application.WorksheetFunction _
      .Match(9, Worksheets(1).Range("A1:A10"), 0)
   MsgBox myVar
End Sub
```

Note Visual Basic functions don't use the **WorksheetFunction** qualifier. A function may have the same name as a Microsoft Excel function and yet work differently. For example, `Application.WorksheetFunction.Log` and `Log` will return different values.

Inserting a Worksheet Function into a Cell

To insert a worksheet function into a cell, you specify the function as the value of the **Formula** property of the corresponding **Range** object. In the following example, the RAND worksheet function (which generates a random number) is assigned to the **Formula** property of range A1:B3 on Sheet1 in the active workbook.

```
Sub InsertFormula()
   Worksheets("Sheet1").Range("A1:B3").Formula = "=RAND()"
End Sub
```

Example of Using Worksheet Functions in Visual Basic

This example uses the worksheet function **Pmt** to calculate a home mortgage loan payment. Notice that this example uses the **InputBox** method instead of the **InputBox** function so that the method can perform type checking. The **Static** statements cause Visual Basic to retain the values of the three variables; these are displayed as default values the next time you run the program.

```
Static loanAmt
Static loanInt
Static loanTerm
loanAmt = Application.InputBox _
   (Prompt:="Loan amount (100,000 for example)", _
      Default:=loanAmt, Type:=1)
loanInt = Application.InputBox _
   (Prompt:="Annual interest rate (8.75 for example)", _
      Default:=loanInt, Type:=1)
loanTerm = Application.InputBox _
   (Prompt:="Term in years (30 for example)", _
      Default:=loanTerm, Type:=1)
payment = Application.WorksheetFunction _
   .Pmt(loanInt / 1200, loanTerm * 12, loanAmt)
MsgBox "Monthly payment is " & Format(payment, "Currency")
```

List of Worksheet Functions Available to Visual Basic

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscListOfWorksheetFunctionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmscListOfWorksheetFunctionsX":1}

A

Acos

Acosh

And

Asin

Asinh

Atan2

Atanh

AveDev

Average

B

BetaDist

BetaInv

BinomDist

C

Ceiling

ChiDist

ChiInv

ChiTest

Choose

Clean

Combin

Confidence

Correl

Cosh

Count

CountA

CountBlank

CountIf

Covar

CritBinom

D

DAverage

Days360

Db

DCount

DCountA

Ddb

Degrees

DevSq

DGet

DMax

DMin

Dollar

DProduct

DStDev

DStDevP

DSum

DVar

DVarP

E

Even

ExponDist

F

Fact

FDist

Find

FindB

FInv

Fisher

FisherInv

Fixed

Floor

Forecast

Frequency

FTest

Fv

G

GammaDist

GammaInv

GammaLn

GeoMean

Growth

H

HarMean

HLookup

HypGeomDist

I

Index

Intercept

Ipmt

Irr

IsErr

IsError

IsLogical

IsNA

IsNonText

IsNumber

Ispmt

IsText

J

K

Kurt

L

Large

LinEst

Ln

Log

Log10

LogEst

LogInv

LogNormDist

Lookup

M

Match

Max

MDeterm

Median

Min

MInverse

MIrr

MMult

Mode

N

NegBinomDist

NormDist

NormInv

NormSDist

NormSInv

NPer

Npv

O

Odd

Or

P

Pearson

Percentile

PercentRank

Permut

Pi

Pmt

Poisson

Power

Ppmt

Prob

Product

Proper

Pv

Q

Quartile

R

Radians

Rank

Rate

Replace

ReplaceB

Rept

Roman

Round

RoundDown

RoundUp

RSq

S

Search

SearchB

Sinh

Skew

Sln

Slope

Small

Standardize

StDev

StDevP

StEyx

Substitute

Subtotal

Sum

SumIf

SumProduct

SumSq

SumX2MY2

SumX2PY2

SumXMY2

Syd

T

Tanh

TDist

Text

TInv

Transpose

Trend

Trim

TrimMean

TTest

U

USDollar

V

Var

VarP

Vdb

VLookup

W

Weekday

Weibull

X

Y

Z

ZTest

This is a worksheet function that's available only in Far East versions of Microsoft Excel.

ActivePane Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproActivePaneC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproActivePaneX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproActivePaneA "}

Returns a **Pane** object that represents the active pane in the window. Read-only.

Remarks

This property can be used only on worksheets and macro sheets.

This property returns a **Pane** object. You must use the **Index** property to obtain the index of the active pane.

ActivePane Property Example

This example activates the next pane of the active window in Book1.xls. You cannot activate the next pane if the panes are frozen. The example must be run from a workbook other than Book1.xls. Before running the example, make sure that Book1.xls has either two or four panes in the active worksheet.

```
Workbooks("BOOK1.XLS").Activate
If not ActiveWindow.FreezePanes Then
   With ActiveWindow
      i = .ActivePane.Index
      If i = .Panes.Count Then
         .Panes(1).Activate
      Else
         .Panes(i+1).Activate
      End If
   End With
End If
```

Arrange Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthArrangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthArrangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthArrangeA "}

Arranges the windows on the screen.

Syntax

expression.**Arrange(*ArrangeStyle, ActiveWorkbook, SyncHorizontal, SyncVertical*)**

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

ArrangeStyle Optional **Variant**. Can be one of the following **XlArrangeStyle** constants.

Constant	Description
xlArrangeStyleTiled	Windows are tiled (the default value).
xlArrangeStyleCascade	Windows are cascaded.
xlArrangeStyleHorizontal	Windows are arranged horizontally.
xlArrangeStyleVertical	Windows are arranged vertically.

ActiveWorkbook Optional **Variant**. **True** to arrange only the visible windows of the active workbook. **False** to arrange all windows. The default value is **False**.

SyncHorizontal Optional **Variant**. Ignored if ***ActiveWorkbook*** is **False** or omitted. **True** to synchronize the windows of the active workbook when scrolling horizontally. **False** to not synchronize the windows. The default value is **False**.

SyncVertical Optional **Variant**. Ignored if ***ActiveWorkbook*** is **False** or omitted. **True** to synchronize the windows of the active workbook when scrolling vertically. **False** to not synchronize the windows. The default value is **False**.

Arrange Method Example

This example tiles all the windows in the application.

```
Application.Windows.Arrange ArrangeStyle:=xlArrangeStyleTiled
```

Cell Error Values

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscCellErrorValuesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmscCellErrorValuesX ":1}

You can insert a cell error value into a cell or test the value of a cell for an error value by using the **CVErr** function. The cell error values can be one of the following **XlCVError** constants.

Constant	Error number	Cell error value
xlErrDiv0	2007	#DIV/0!
xlErrNA	2042	#N/A
xlErrName	2029	#NAME?
xlErrNull	2000	#NULL!
xlErrNum	2036	#NUM!
xlErrRef	2023	#REF!
xlErrValue	2015	#VALUE!

Cell Error Values Example

This example inserts the seven cell error values into cells A1:A7 on Sheet1.

```
myArray = Array(xlErrDiv0, xlErrNA, xlErrName, xlErrNull, _
   xlErrNum, xlErrRef, xlErrValue)
For i = 1 To 7
   Worksheets("Sheet1").Cells(i, 1).Value = CVErr(myArray(i - 1))
Next i
```

This example displays a message if the active cell on Sheet1 contains a cell error value. You can use this example as a framework for a cell-error-value error handler.

```
Worksheets("Sheet1").Activate
If IsError(ActiveCell.Value) Then
   errval = ActiveCell.Value
   Select Case errval
      Case CVErr(xlErrDiv0)
         MsgBox "#DIV/0! error"
      Case CVErr(xlErrNA)
         MsgBox "#N/A error"
      Case CVErr(xlErrName)
         MsgBox "#NAME? error"
      Case CVErr(xlErrNull)
         MsgBox "#NULL! error"
      Case CVErr(xlErrNum)
         MsgBox "#NUM! error"
      Case CVErr(xlErrRef)
         MsgBox "#REF! error"
      Case CVErr(xlErrValue)
         MsgBox "#VALUE! error"
      Case Else
         MsgBox "This should never happen!!"
   End Select
End If
```

Date1904 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDate1904C "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDate1904X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDate1904A "}

True if the workbook uses the 1904 date system. Read/write **Boolean**.

Date1904 Property Example

This example causes Microsoft Excel to use the 1904 date system for the active workbook.

```
ActiveWorkbook.Date1904 = True
```

DeleteNumberFormat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDeleteNumberFormatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDeleteNumberFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDeleteNumberFormatA "}

Deletes a custom number format from the workbook.

Syntax

expression.**DeleteNumberFormat(*NumberFormat*)**

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

NumberFormat Required **String**. Names the number format to be deleted.

DeleteNumberFormat Method Example

This example deletes the number format "000-00-0000" from the active workbook.

```
ActiveWorkbook.DeleteNumberFormat("000-00-0000")
```

DisplayDrawingObjects Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayDrawingObjectsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayDrawingObjectsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayDrawingObjectsA "}

Returns or sets how shapes are displayed. Read/write **Long**.

Can be one of the following **XlDisplayShapes** constants.

Constant	Description
xlDisplayShapes	Show all shapes.
xlPlaceholders	Show only placeholders.
xlHide	Hide all shapes.

DisplayDrawingObjects Property Example

This example hides all the shapes in the active workbook.

```
ActiveWorkbook.DisplayDrawingObjects = xlHide
```

DisplayFormulas Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayFormulasC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayFormulasX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayFormulasA "}

True if the window is displaying formulas, **False** if the window is displaying values. Read/write **Boolean**.

Remarks

This property applies only to worksheets and macro sheets.

DisplayFormulas Property Example

This example changes the active window in Book1.xls to display formulas.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.DisplayFormulas = True
```

DisplayGridlines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayGridlinesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayGridlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayGridlinesA "}

True if gridlines are displayed. Read/write **Boolean**.

Remarks

This property applies only to worksheets and macro sheets.

This property affects only displayed gridlines. Use the **PrintGridlines** property to control the printing of gridlines.

DisplayGridlines Property Example

This example toggles the display of gridlines in the active window in Book1.xls.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.DisplayGridlines = Not(ActiveWindow.DisplayGridlines)
```

DisplayHeadings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayHeadingsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayHeadingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayHeadingsA "}

True if both row and column headings are displayed, **False** if there are no headings displayed. Read/write **Boolean**.

Remarks

This property applies only to worksheets and macro sheets.

This property affects only displayed headings. Use the **PrintHeadings** property to control the printing of headings.

DisplayHeadings Property Example

This example turns off the display of row and column headings in the active window in Book1.xls.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.DisplayHeadings = False
```

DisplayOutline Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayOutlineC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayOutlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayOutlineA "}

True if outline symbols are displayed. Read/write **Boolean**.

Remarks

This property applies only to worksheets and macro sheets.

DisplayOutline Property Example

This example displays outline symbols for the active window in Book1.xls.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.DisplayOutline = True
```

DisplayZeros Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayZerosC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayZerosX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayZerosA "}

True if zero values are displayed. Read/write **Boolean**.

Remarks

This property applies only to worksheets and macro sheets.

DisplayZeros Property Example

This example sets the active window in Book1.xls to display zero values.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.DisplayZeros = True
```

FreezePanes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFreezePanesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFreezePanesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFreezePanesA "}

True if split panes are frozen. Read/write **Boolean**.

Remarks

It's possible for **FreezePanes** to be **True** and **Split** to be **False**, or vice versa.

This property applies only to worksheets and macro sheets.

FreezePanes Property Example

This example freezes split panes in the active window in Book1.xls.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.FreezePanes = True
```

GridlineColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGridlineColorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGridlineColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGridlineColorA "}

Returns or sets the gridline color as an RGB value. Read/write **Long**.

GridlineColor Property Example

This example sets the gridline color in the active window in Book1.xls to red.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.GridlineColor = RGB(255,0,0)
```

IncludeAlignment Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIncludeAlignmentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIncludeAlignmentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIncludeAlignmentA "}

True if the style includes the **AddIndent**, **HorizontalAlignment**, **VerticalAlignment**, **WrapText**, and **Orientation** properties. Read/write **Boolean**.

IncludeAlignment Property Example

This example sets the style attached to cell A1 on Sheet1 to include alignment format.

```
Worksheets("Sheet1").Range("A1").Style.IncludeAlignment = True
```

IncludeBorder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIncludeBorderC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIncludeBorderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIncludeBorderA "}

True if the style includes the **Color**, **ColorIndex**, **LineStyle**, and **Weight** border properties. Read/write **Boolean**.

IncludeBorder Property Example

This example sets the style attached to cell A1 on Sheet1 to include border format.

```
Worksheets("Sheet1").Range("A1").Style.IncludeBorder = True
```

IncludeFont Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIncludeFontC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIncludeFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIncludeFontA "}

True if the style includes the **Background**, **Bold**, **Color**, **ColorIndex**, **FontStyle**, **Italic**, **Name**, **OutlineFont**, **Shadow**, **Size**, **Strikethrough**, **Subscript**, **Superscript**, and **Underline** font properties. Read/write **Boolean**.

IncludeFont Property Example

This example sets the style attached to cell A1 on Sheet1 to include font format.

```
Worksheets("Sheet1").Range("A1").Style.IncludeFont = True
```

IncludeNumber Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIncludeNumberC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIncludeNumberX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIncludeNumberA "}

True if the style includes the **NumberFormat** property. Read/write **Boolean**

IncludeNumber Property Example

This example sets the style attached to cell A1 on Sheet1 to include number format.

```
Worksheets("Sheet1").Range("A1").Style.IncludeNumber = True
```

IncludePatterns Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIncludePatternsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIncludePatternsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIncludePatternsA "}

True if the style includes the **Color**, **ColorIndex**, **InvertIfNegative**, **Pattern**, **PatternColor**, and **PatternColorIndex** interior properties. Read/write **Boolean**.

IncludePatterns Property Example

This example sets the style attached to cell A1 on Sheet1 to include pattern format.

```
Worksheets("Sheet1").Range("A1").Style.IncludePatterns = True
```

IncludeProtection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIncludeProtectionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIncludeProtectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIncludeProtectionA "}

True if the style includes the **FormulaHidden** and **Locked** protection properties. Read/write **Boolean**.

IncludeProtection Property Example

This example sets the style attached to cell A1 on Sheet1 to include protection format.

```
Worksheets("Sheet1").Range("A1").Style.IncludeProtection = True
```

PrecisionAsDisplayed Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrecisionAsDisplayedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrecisionAsDisplayedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrecisionAsDisplayedA "}

True if calculations in this workbook will be done using only the precision of the numbers as they're displayed. Read/write **Boolean**.

PrecisionAsDisplayed Property Example

This example causes calculations in the active workbook to use only the precision of the numbers as they're displayed.

```
ActiveWorkbook.PrecisionAsDisplayed = True
```

SaveLinkValues Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSaveLinkValuesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSaveLinkValuesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSaveLinkValuesA "}

True if Microsoft Excel saves external link values with the workbook. Read/write **Boolean**.

SaveLinkValues Property Example

This example causes Microsoft Excel to save external link values with the active workbook.

```
ActiveWorkbook.SaveLinkValues = True
```

ShowDataForm Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowDataFormC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowDataFormX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowDataFormA "}

Displays the data form associated with the worksheet.

Syntax

expression.**ShowDataForm**

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

Remarks

The macro pauses while you're using the data form. When you close the data form, the macro resumes at the line following the **ShowDataForm** method.

This method runs the custom data form, if one exists.

ShowDataForm Method Example

This example displays the data form for Sheet1.

```
Worksheets(1).ShowDataForm
```

SpecialCells Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSpecialCellsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSpecialCellsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSpecialCellsA "}

Returns a **Range** object that represents all the cells that match the specified type and value.

Syntax

expression.**SpecialCells(*Type, Value*)**

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

Type Required **Long**. The cells to include. Can be one of the following **XlCellType** constants.

Constant	Description
xlCellTypeNotes	Cells containing notes
xlCellTypeConstants	Cells containing constants
xlCellTypeFormulas	Cells containing formulas
xlCellTypeBlanks	Empty cells
xlCellTypeLastCell	The last cell in the used range
xlCellTypeVisible	All visible cells

Value Optional **Variant**. If ***Type*** is either **xlCellTypeConstants** or **xlCellTypeFormulas**, this argument is used to determine which types of cells to include in the result. These values can be added together to return more than one type. The default is to select all constants or formulas, no matter what the type. Can be one of the following **XlSpecialCellsValues** constants: **xlErrors**, **xlLogical**, **xlNumbers**, **xlTextValues**, **xlAllFormatConditions**, or **xlSameFormatConditions**.

SpecialCells Method Example

This example selects the last cell in the used range of Sheet1.

```
Worksheets("Sheet1").Activate
ActiveSheet.Cells.SpecialCells(xlCellTypeLastCell).Activate
```

Split Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSplitC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSplitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSplitA "}

True if the window is split. Read/write **Boolean**.

Remarks

It's possible for **FreezePanes** to be **True** and **Split** to be **False**, or vice versa.

This property applies only to worksheets and macro sheets.

Split Property Example

This example splits the active window in Book1.xls at cell B2, without freezing panes. This causes the **Split** property to return **True**.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
With ActiveWindow
    .SplitColumn = 2
    .SplitRow = 2
End With
```

This example illustrates two ways of removing the split added by the preceding example.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.Split = False          'method one
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.SplitColumn = 0        'method two
ActiveWindow.SplitRow = 0
```

This example removes the window split. Before you can remove the split, you must set **FreezePanes** to **False** to remove frozen panes.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
With ActiveWindow
    .FreezePanes = False
    .Split = False
End With
```

SplitColumn Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSplitColumnC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSplitColumnX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSplitColumnA "}

Returns or sets the column number where the window is split into panes (the number of columns to the left of the split line). Read/write **Long**.

SplitColumn Property Example

This example splits the window and leaves 1.5 columns to the left of the split line.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.SplitColumn = 1.5
```

SplitHorizontal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSplitHorizontalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSplitHorizontalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSplitHorizontalA "}

Returns or sets the location of the horizontal window split, in points. Read/write **Double**.

SplitHorizontal Property Example

This example sets the horizontal split for the active window to 216 points (3 inches).

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.SplitHorizontal = 216
```

SplitRow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSplitRowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSplitRowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSplitRowA "}

Returns or sets the row number where the window is split into panes (the number of rows above the split). Read/write **Long**.

SplitRow Property Example

This example splits the active window so that there are 10 rows above the split line.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.SplitRow = 10
```

SplitVertical Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSplitVerticalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSplitVerticalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSplitVerticalA "}

Returns or sets the location of the vertical window split, in points. Read/write **Double**.

SplitVertical Property Example

This example sets the vertical split for the active window to 216 points (3 inches).

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.SplitVertical = 216
```

StandardHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStandardHeightC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStandardHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStandardHeightA "}

Returns the standard (default) height of all the rows in the worksheet, in points. Read-only **Double**.

StandardHeight Property Example

This example sets the height of row one on Sheet1 to the standard height.

```
Worksheets("Sheet1").Rows(1).RowHeight = _
    Worksheets("Sheet1").StandardHeight
```

StandardWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproStandardWidthC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproStandardWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproStandardWidthA "}

Returns or sets the standard (default) width of all the columns in the worksheet. Read/write **Double**.

Remarks

One unit of column width is equal to the width of one character in the Normal style. For proportional fonts, the width of the character 0 (zero) is used.

StandardWidth Property Example

This example sets the width of column one on Sheet1 to the standard width.

```
Worksheets("Sheet1").Columns(1).ColumnWidth = _
    Worksheets("Sheet1").StandardWidth
```

TransitionExpEval Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTransitionExpEvalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransitionExpEvalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransitionExpEvalA "}

True if Microsoft Excel uses Lotus 1-2-3 expression evaluation rules for the worksheet. Read/write **Boolean**.

TransitionExpEval Property Example

This example causes Microsoft Excel to use Lotus 1-2-3 expression evaluation rules for Sheet1.

```
Worksheets("Sheet1").TransitionExpEval = True
```

TransitionFormEntry Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTransitionFormEntryC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransitionFormEntryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransitionFormEntryA "}

True if Microsoft Excel uses Lotus 1-2-3 formula entry rules for the worksheet. Read/write **Boolean**.

Remarks

This property isn't available on the Macintosh.

TransitionFormEntry Property Example

This example causes Microsoft Excel to use Lotus 1-2-3 formula entry rules for Sheet1.

```
Worksheets("Sheet1").TransitionFormEntry = True
```

UpdateRemoteReferences Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUpdateRemoteReferencesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUpdateRemoteReferencesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUpdateRemoteReferencesA "}

True if Microsoft Excel updates remote references in for the workbook. Read/write **Boolean**.

UpdateRemoteReferences Property Example

This example causes remote references to be updated in the active workbook.

```
ActiveWorkbook.UpdateRemoteReferences = True
```

UsedRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUsedRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUsedRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUsedRangeA "}

Returns a **Range** object that represents the used range on the specified worksheet. Read-only.

UsedRange Property Example

This example selects the used range on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveSheet.UsedRange.Select
```

AutoComplete Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoCompleteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoCompleteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoCompleteA "}

Returns an AutoComplete match from the list. If there's no AutoComplete match or if more than one entry in the list matches the string to complete, this method returns an empty string.

Syntax

expression.**AutoComplete(*String*)**

expression Required. An expression that returns a **Range** object (must be a single cell).

String Required **String**. The string to complete.

Remarks

This method works even if the AutoComplete feature is disabled.

AutoComplete Method Example

This example returns the AutoComplete match for the string segment "Ap." An AutoComplete match is made if the column containing cell A5 contains a contiguous list and one of the entries in the list contains a match for the string.

```
s = Worksheets(1).Range("A5").AutoComplete("Ap")
If Len(s) > 0 Then
   MsgBox "Completes to " & s
Else
   MsgBox "Has no completion"
End If
```

Container Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproContainerC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproContainerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproContainerA "}

Returns the object that contains the specified embedded workbook. Read-only **Object**.

Remarks

Use this property with a contained workbook to return the container object. If the container doesn't support OLE Automation or the workbook isn't embedded, this property fails.

Container Property Example

This example hides the second section in the binder that contains the active Microsoft Excel workbook and then sets the value of cell A1 to 345.67. In this example, the binder is Binder1.obd.

```
Set myBinder = GetObject("Binder1.obd", "Office.Binder")
Set myWorkbook = myBinder.Sections(1).Object
With myWorkbook
   .Container.Sections(2).Visible = False
   .Sheets(1).Cells(1, 1).Value = 345.67
End With
```

EnableAnimations Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableAnimationsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableAnimationsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableAnimationsA "}

True if animated insertion and deletion is enabled. Read/write **Boolean**.

Remarks

When animation is enabled, inserted worksheet rows and columns appear slowly, and deleted worksheet rows and columns disappear slowly.

EnableAnimations Property Example

This example turns off animated insertion and deletion.

```
Application.EnableAnimations = False
```

EnableAutoComplete Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableAutoCompleteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableAutoCompleteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableAutoCompleteA "}

True if the AutoComplete feature is enabled. Read/write **Boolean**.

EnableAutoComplete Property Example

This example enables the AutoComplete feature.

```
Application.EnableAutoComplete = True
```

ListHeaderRows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproListHeaderRowsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproListHeaderRowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproListHeaderRowsA "}

Returns the number of header rows for the specified range. Read-only **Long**.

Remarks

Before you use this property, use the **CurrentRegion** property to find the boundaries of the range.

ListHeaderRows Property Example

This example sets the `rTbl` variable to the range represented by the current region for the active cell, not including any header rows.

```
Set rTbl = ActiveCell.CurrentRegion
' remove the headers from the range
iHdrRows = rTbl.ListHeaderRows
If iHdrRows > 0 Then
   ' resize the range minus n rows
   Set rTbl = rTbl.Resize(rTbl.Rows.Count - iHdrRows)
   ' and then move the resized range down to
   ' get to the first non-header row
   Set rTbl = rTbl.Offset(iHdrRows)
End If
```

MoveAfterReturnDirection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMoveAfterReturnDirectionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMoveAfterReturnDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMoveAfterReturnDirectionA "}

Returns or sets the direction in which the active cell is moved when the user presses ENTER. Can be one of the following **XlDirection** constants: **xlToLeft**, **xlToRight**, **xlUp**, or **xlDown**. Read/write **Long**.

Remarks

If the **MoveAfterReturn** property is **False**, the selection doesn't move at all, regardless of how the **MoveAfterReturnDirection** property is set.

MoveAfterReturnDirection Property Example

This example causes the active cell to move to the right when the user presses ENTER.

```
Application.MoveAfterReturn = True
Application.MoveAfterReturnDirection = xlToRight
```

NetworkTemplatesPath Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNetworkTemplatesPathC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNetworkTemplatesPathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNetworkTemplatesPathA "}

Returns the network path where templates are stored. If the network path doesn't exist, this property returns an empty string. Read-only **String**.

NetworkTemplatesPath Property Example

This example displays the network path where templates are stored.

```
Msgbox Application.NetworkTemplatesPath
```

ProtectionMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectionModeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectionModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectionModeA "}

True if user-interface-only protection is turned on. To turn on user interface protection, use the **Protect** method with the ***UserInterfaceOnly*** argument set to **True**. Read-only **Boolean**.

ProtectionMode Property Example

This example displays the status of the **ProtectionMode** property.

```
MsgBox ActiveSheet.ProtectionMode
```

RangeSelection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRangeSelectionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRangeSelectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRangeSelectionA "}

Returns a **Range** object that represents the selected cells on the worksheet in the specified window even if a graphic object is active or selected on the worksheet. Read-only.

Remarks

When a graphic object is selected on a worksheet, the **Selection** property returns the graphic object instead of a **Range** object; the **RangeSelection** property returns the range of cells that was selected before the graphic object was selected.

This property and the **Selection** property return identical values when a range (not a graphic object) is selected on the worksheet.

If the active sheet in the specified window isn't a worksheet, this property fails.

RangeSelection Property Example

This example displays the address of the selected cells on the worksheet in the active window.

```
MsgBox ActiveWindow.RangeSelection.Address
```

SetBackgroundPicture Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetBackgroundPictureC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetBackgroundPictureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetBackgroundPictureA "}

Sets the background graphic for a worksheet or chart.

Syntax

expression.**SetBackgroundPicture(*FileName*)**

expression Required. An expression that returns a **Worksheet** or **Chart** object.

FileName Required **String**. The name of the graphic file.

SetBackgroundPicture Method Example

This example sets the background graphic for worksheet one.

```
Worksheets(1).SetBackgroundPicture "c:\graphics\watrmark.bmp"
```

TemplatesPath Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTemplatesPathC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTemplatesPathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTemplatesPathA "}

Returns the local path where templates are stored. Read-only **String**.

TemplatesPath Property Example

This example returns the local path where templates are stored.

```
Msgbox Application.TemplatesPath
```

CreateBackup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCreateBackupC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCreateBackupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCreateBackupA "}

True if a backup file is created when this file is saved. Read-only **Boolean**.

CreateBackup Property Example

This example displays a message if a backup file is created when the active workbook is saved.

```
If ActiveWorkbook.CreateBackup = True Then
   MsgBox "Remember, there is a backup copy of this workbook"
End If
```

FileFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFileFormatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFileFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFileFormatA "}

Returns the file format and/or type of the workbook. Read-only **Long**.

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

xlAddIn
xlCSV
xlCSVMac
xlCSVMSDOS
xlCSVWindows
xlDBF2
xlDBF3
xlDBF4
xlDIF
xlExcel2
xlExcel2FarEast
xlExcel3
xlExcel4
xlExcel5
xlExcel4Workbook
xlIntlAddIn
xlIntlMacro
xlWorkbookNormal
xlSYLK
xlTemplate
xlCurrentPlatformText
xlTextMac
xlTextMSDOS
xlTextPrinter
xlTextWindows
xlWJ2WD1
xlWK1
xlWK1ALL
xlWK1FMT
xlWK3
xlWK4
xlWK3FM3
xlWKS
xlWorks2FarEast
xlWQ1
xlWJ3
xlWJ3FJ3

Remarks

The following additional formats are available in the Far East version of Microsoft Excel: **xlWJ2WD1**, **xlExcel2FarEast**, and **xlWorks2FarEast**.

FileFormat Property Example

This example saves the active workbook in Normal file format if its current file format is WK3.

```
If ActiveWorkbook.FileFormat = xlWK3 Then
   ActiveWorkbook.SaveAs fileFormat:=xlNormal
End If
```

HasPassword Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasPasswordC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasPasswordX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasPasswordA "}

True if the workbook has a protection password. Read-only **Boolean**.

Remarks

You can assign a protection password to a workbook by using the **SaveAs** method.

HasPassword Property Example

This example displays a message if the active workbook has a protection password.

```
If ActiveWorkbook.HasPassword = True Then
   MsgBox "Remember to obtain the workbook password" & Chr(13) & _
      " from the Network Administrator."
End If
```

NewWindow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthNewWindowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthNewWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthNewWindowA "}

Creates a new window or a copy of the specified window.

Syntax

expression.**NewWindow**

expression Required. An expression that returns a **Window** or **Workbook** object.

NewWindow Method Example

This example creates a new window for the active workbook.

```
ActiveWorkbook.NewWindow
```

ProtectStructure Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectStructureC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectStructureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectStructureA "}

True if the order of the sheets in the workbook is protected. Read-only **Boolean**.

ProtectStructure Property Example

This example displays a message if the order of the sheets in the active workbook is protected.

```
If ActiveWorkbook.ProtectStructure = True Then
   MsgBox "Remember, you cannot delete, add, or change " & Chr(13) & _
      "the location of any sheets in this workbook."
End If
```

ProtectWindows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectWindowsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectWindowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectWindowsA "}

True if the windows of the workbook are protected. Read-only **Boolean**.

ProtectWindows Property Example

This example displays a message if the windows in the active workbook are protected.

```
If ActiveWorkbook.ProtectWindows = True Then
   MsgBox "Remember, you cannot rearrange any window in this workbook."
End If
```

ReadOnly Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReadOnlyC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReadOnlyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReadOnlyA "}

True if the workbook has been opened as read-only. Read-only **Boolean**.

ReadOnly Property Example

If the active workbook is read-only, this example saves it as Newfile.xls.

```
If ActiveWorkbook.ReadOnly Then
   ActiveWorkbook.SaveAs fileName:="NEWFILE.XLS"
End If
```

ReadOnlyRecommended Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReadOnlyRecommendedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReadOnlyRecommendedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReadOnlyRecommendedA "}

True if the workbook was saved as read-only recommended. Read-only **Boolean**.

Remarks

When you open a workbook that was saved as read-only recommended, Microsoft Excel displays a message recommending that you open the workbook as read-only.

Use the **SaveAs** method to change this property.

ReadOnlyRecommended Property Example

This example displays a message if the active workbook is saved as read-only recommended.

```
If ActiveWorkbook.ReadOnlyRecommended = True Then
   MsgBox "This workbook is saved as read-only recommended"
End If
```

Saved Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSavedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSavedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSavedA "}

True if no changes have been made to the specified workbook since it was last saved. Read/write **Boolean**.

Remarks

If a workbook has never been saved, its **Path** property returns an empty string ("").

You can set this property to **True** if you want to close a modified workbook without either saving it or being prompted to save it.

Saved Property Example

This example displays a message if the active workbook contains unsaved changes.

```
If Not ActiveWorkbook.Saved Then
   MsgBox "This workbook contains unsaved changes."
End If
```

This example closes the workbook that contains the example code and discards any changes to the workbook by setting the **Saved** property to **True**.

```
ThisWorkbook.Saved = True
ThisWorkbook.Close
```

WriteReserved Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWriteReservedC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWriteReservedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWriteReservedA "}

True if the workbook is write-reserved. Read-only **Boolean**.

Remarks

Use the **SaveAs** method to set this property.

WriteReserved Property Example

If the active workbook is write-reserved, this example displays a message that contains the name of the user who saved the workbook as write-reserved.

```
With ActiveWorkbook
   If .WriteReserved = True Then
      MsgBox "Please contact " & .WriteReservedBy & Chr(13) & _
         " if you need to insert data in this workbook."
   End If
End With
```

WriteReservedBy Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWriteReservedByC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWriteReservedByX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWriteReservedByA "}

Returns the name of the user who currently has write permission for the workbook. Read-only **String**.

WriteReservedBy Property Example

If the active workbook is write-reserved, this example displays a message that contains the name of the user who saved the workbook as write-reserved.

```
With ActiveWorkbook
   If .WriteReserved = True Then
      MsgBox "Please contact " & .WriteReservedBy & Chr(13) & _
         " if you need to insert data in this workbook."
   End If
End With
```

Clone Method Example (Microsoft Excel)

This example displays a custom dialog box that contains the lists of data from the CONTACTS and CUSTMR_ID fields in the Customer recordset of the Nwindex.mdb database.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database
Dim rs1 As Recordset, rs2 As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs1 = db.OpenRecordset("SELECT * FROM Customer" _
   & " WHERE [REGION] = 'WA' ORDER BY [CUSTMR_ID];")
Set theDialog = DialogSheets.Add
Set list1 = theDialog.ListBoxes.Add(78, 42, 84, 80)
Set list2 = theDialog.ListBoxes.Add(183, 42, 84, 80)
Set rs2 = rs1.Clone()
rs2.MoveFirst
Do Until rs1.EOF
   list1.AddItem (rs1.Fields("CONTACT").Value)
   rs1.MoveNext
Loop
Do Until rs2.EOF
   list2.AddItem (rs2.Fields("CUSTMR_ID").Value)
   rs2.MoveNext
Loop
rs1.Close
rs2.Close
db.Close
```

Close Method Example (Microsoft Excel)

This example opens the Customer recordset of the Nwindex.mdb database, counts how many records are available, and enters this number on Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Customer")
Set resultsSheet = Sheets("Sheet1")
resultsSheet.Activate
With resultsSheet.Cells(1, 1)
    .Value = "Records in " & rs.Name & " table:"
    .Font.Bold = True
    .EntireColumn.AutoFit
End With
rs.MoveLast
resultsSheet.Cells(1, 2).Value = rs.RecordCount
rs.Close
db.Close
```

CreateDatabase, CreateTableDef, Append Methods Example (Microsoft Excel)

This example creates a new database, Nwindex.mdb. The example attaches two tables from the C:\Program Files\Common Files\Microsoft Shared\MSquery folder to the database. (In Windows NT, the two tables are located in the C:\Windows\Msapps\Msquery folder.)

```
Dim nWindEx As Database, customerTable As TableDef, supplierTable As
TableDef
Dim dataSource As String
dataSource = _
"dBASE IV;DATABASE=C:\Program Files\Common Files\Microsoft Shared\MSquery"
appPath = Application.Path
Set nWindEx = Workspaces(0).CreateDatabase(Application.Path _
   & "\NWINDEX.MDB", dbLangGeneral)
Set customerTable = nWindEx.CreateTableDef("Customer")
customerTable.Connect = dataSource
customerTable.SourceTableName = "Customer"
nWindEx.TableDefs.Append customerTable
Set supplierTable = nWindEx.CreateTableDef("Supplier")
supplierTable.Connect = dataSource
supplierTable.SourceTableName = "Supplier"
nWindEx.TableDefs.Append supplierTable
MsgBox "The database " & nWindEx.Name & " has been created."
nWindEx.Close
```

Delete Method Example (Microsoft Excel)

This example adds a new record to Product.dbf (a dBASE IV table located in the C:\Program Files\Common Files\Microsoft Shared\MSquery folder) and then deletes the record. (In Windows NT, Product.dbf is located in the C:\Windows\Msapps\Msquery folder.)

```
Dim db As Database, rs As Recordset
Sheets("Sheet1").Activate
Set theID = ActiveSheet.Cells(1, 2)
Set theCategory = ActiveSheet.Cells(2, 2)
theID.Value = 200
theCategory.Value = "BEVR"
Set db = OpenDatabase("C:\Program Files\Common Files\Microsoft Shared\
MSquery", False, _
   False, "dBASE IV")
Set rs = db.OpenRecordset("PRODUCT.DBF", dbOpenTable)
rs.AddNew
rs("PRODUCT_ID") = theID.Value
rs("CATEGORY") = theCategory.Value
rs.Update
MsgBox "The new record has been created with " & theID.Value _
   & " and " & theCategory.Value
rs.Move 0, rs.LastModified
rs.Delete
MsgBox "The record you just created has been deleted"
rs.Close
db.Close
```

OpenRecordset Method Example (Microsoft Excel)

This example displays a custom dialog box that contains a list of all the available recordsets in the Nwindex.mdb database. The example opens a new recordset based on the recordset selected by the user and then copies the records onto Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs1 As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Application.ScreenUpdating = False
Set theDialog = DialogSheets.Add
Set list1 = theDialog.ListBoxes.Add(78, 42, 84, 80)
Set label1 = theDialog.Labels.Add(78, 125, 240, 25)
label1.Text = "Recordsets for " & db.Name
i = 0
Do Until i = db.TableDefs.Count
   list1.AddItem (db.TableDefs(i).Name)
   i = i + 1
Loop
Sheets("Sheet1").Activate
Application.ScreenUpdating = True
If theDialog.Show = True Then
   If list1.Value = 0 Then
      MsgBox "You have not selected an item from the list."
   Else
      Set rs1 = db.OpenRecordset(db.TableDefs(list1.Value - 1).Name)
      ActiveCell.CopyFromRecordset rs1
   End If
End If
rs1.Close
db.Close
```

GetRows Method Example (Microsoft Excel)

This example copies selected records from the Customer recordset in the Nwindex.mdb database to Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset
Dim data As Variant
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("SELECT CUSTMR_ID, CONTACT FROM Customer;")
data = rs.GetRows(6)
Sheets("Sheet1").Activate
For r = 1 to UBound(data, 2) + 1
   For c = 1 to 2
      Cells(r, c).Value = data(c - 1, r - 1)
   Next
Next
rs.Close
db.Close
```

Move Method Example (Microsoft Excel)

This example prompts the user to select a record number. The example then copies the selected record from the Customer recordset in the Nwindex.mdb database to Sheet1.

```
Dim db As Database, rs As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("SELECT [CUSTMR_ID], [CONTACT], [REGION] " _
   & "FROM Customer")
Sheets("Sheet1").Activate
aDistance = Application.InputBox("What record # you want to copy", _
   Type:=2)
If aDistance = False Then  ' user cancelled InputBox
   Exit Sub
End If
rs.MoveFirst
rs.Move aDistance
For i = 0 To 2
   ActiveCell.Offset(, i).Value = rs.Fields(i).Value
Next
rs.Close
db.Close
```

MoveNext Method Example (Microsoft Excel)

This example replaces values in the CON_TITLE field of the records in the Customer recordset in the Nwindex.mdb database, and then it displays the number of replacements that were made.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset, sQLText As String
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
sQLText = "SELECT * FROM Customer WHERE " _
   & "CON_TITLE = 'Sales Representative';"
Set rs = db.OpenRecordset(sQLText)
i = 0
Do Until rs.EOF
   With rs
      .Edit
      .Fields("CON_TITLE").Value = "Account Executive"
      .Update
      .MoveNext
   End With
   i = i + 1
Loop
MsgBox i & " replacements were made."
rs.Close
db.Close
```

OpenDatabase Method Example (Microsoft Excel)

This example displays a custom dialog box that contains a list of all the databases with the file name extension .mdb that are located in the Microsoft Excel folder, and then it opens the database selected by the user.

```
Dim a(100), db As Database
i = 0
ChDrive "C"
ChDir Application.Path
a(i) = Dir("*.MDB")
If a(i) = "" Then
   MsgBox "You have no databases in the Microsoft Excel folder"
   Exit Sub
End If
Do
   i = i + 1
   a(i) = Dir()
Loop Until a(i) = ""
Set theDialog = DialogSheets.Add
Set list1 = theDialog.ListBoxes.Add(78, 42, 84, 80)
For counter = 0 To i - 1
   list1.AddItem a(counter)
Next
Application.ScreenUpdating = True
theDialog.Show
Set db = Workspaces(0).OpenDatabase(a(list1.Value - 1))
MsgBox "The " & db.Name & " database is now open"
' use database here
db.Close
```

Seek Method Example (Microsoft Excel)

This example opens Product.dbf (a dBASE IV table located in the C:\Program Files\Common Files\Microsoft Shared\MSquery folder), locates a record, and then copies the the record's values into cells B2:C2 on Sheet1. (In Windows NT, Product.dbf is located in the C:\Windows\Msapps\Msquery folder.)

```
Dim db As Database, rs As Recordset
Sheets("Sheet1").Activate
Set db = OpenDatabase("C:\Program Files\Common Files\Microsoft Shared\
MSquery", _
      False, False, "dBASE IV")
Set rs = db.OpenRecordset("PRODUCT.DBF", dbOpenTable)
rs.Index = "PRODUCT"
rs.Seek "=", "1"
If rs.NoMatch Then
   MsgBox "Couldn't find any records"
Else
   ActiveSheet.Cells(2, 2) = rs.Fields("CATEGORY").Value
   ActiveSheet.Cells(3, 2) = rs.Fields("PROD_NAME").Value
End If
rs.Close
db.Close
```

Update Method Example (Microsoft Excel)

This example opens Product.dbf (a dBASE IV table located in the C:\Program Files\Common Files\Microsoft Shared\MSquery folder), finds a record with the PRODUCT value 1, and then sets the CATEGORY field to the value in cell B2 on Sheet1. (In Windows NT, Product.dbf is located in the C:\Windows\Msapps\Msquery folder.)

```
Dim db As Database, rs As Recordset, categoryCell As Range
Sheets("Sheet1").Activate
Set categoryCell = ActiveSheet.Cells(2, 2)
categoryCell.Value = "BEVR"
Set db = OpenDatabase("C:\Program Files\Common Files\Microsoft Shared\
MSquery", _
   False, False, "dBASE IV")
Set rs = db.OpenRecordset("PRODUCT.DBF", dbOpenTable)
With rs
   .Index = "PRODUCT"
   .Seek "=", "1"
   .Edit
   .Fields("CATEGORY").Value = categoryCell.Value
   .Update
End With
MsgBox "The field has been updated with " & categoryCell.Value
rs.Close
db.Close
```

AbsolutePosition Property Example (Microsoft Excel)

This example prompts the user for a record number. The example uses this number to move to a record in the Customer recordset in the Nwindex.mdb database, and then it copies the values for three specified fields to Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset
Sheets("Sheet1").Activate
recordNumber = Application.InputBox(Prompt:="Record number to copy " _
   & "to Sheet1", Title:="Record to copy", Type:=1)
If recordNumber = False Then ' user cancelled InputBox
   Exit Sub
End If
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Customer", dbOpenSnapshot)
rs.MoveLast
If rs.RecordCount > recordNumber Then
   rs.AbsolutePosition = recordNumber
   ActiveCell.Value = rs.Fields("CONTACT").Value
   ActiveCell.Offset(, 1).Value = rs.Fields("ADDRESS").Value
   ActiveCell.Offset(, 2).Value = rs.Fields("CITY").Value
Else
   MsgBox "The record #" & recordNumber & " doesn't exist."
End If
rs.Close
db.Close
```

Bookmark, Bookmarkable Properties Example (Microsoft Excel)

This example prompts the user for a two-letter abbreviation for a state. The example uses this value to find up to 101 matching records in the Customer recordset in the Nwindex.mdb database. It then marks each record with a bookmark and copies the values of the first and third fields to Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim Found(100)
i = 0
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Customer")
Sheets("Sheet1").Activate
regionWanted = Application.InputBox("What state do you want data from?", _
   "Specify two letters (e.g. 'WA')", Type:=2)
If regionWanted = False Then ' user cancelled InputBox
   Exit Sub
End If
criteria = "[REGION] = '" & regionWanted & "'"
rs.FindFirst criteria
If rs.NoMatch Or rs.Bookmarkable = False Then
   MsgBox "No records for this state"
   Exit Sub
Else
   Do Until rs.NoMatch = True
      i = i + 1
      Found(i) = rs.Bookmark
      rs.FindNext criteria
   Loop
End If
For n = 1 To i
   rs.Bookmark = Found(n)
   Cells(n + 1, 1).Value = rs.fields(0).Value
   Cells(n + 1, 2).Value = rs.fields(2).Value
Next
MsgBox "There are " & i & " records from this region"
rs.Close
db.Close
```

Connect, SourceTableName Properties Example (Microsoft Excel)

This example attaches the table Product.dbf (a dBASE IV table located in the C:\Program Files\Common Files\Microsoft Shared\MSquery folder) to the Nwindex.mdb database. (In Windows NT, Product.dbf is located in the C:\Windows\Msapps\Msquery folder.)

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim nWindEx As Database, tDef As TableDef
Dim dataSource As String
dataSource = _
   "dBASE IV;DATABASE=C:\Program Files\Common Files\Microsoft Shared\
MSquery"
Set nWindEx = Workspaces(0).OpenDatabase(Application.Path _
   & "\NWINDEX.MDB")
Set tDef = nWindEx.CreateTableDef("Product")
tDef.Connect = dataSource
tDef.SourceTableName = "Product"
nWindEx.TableDefs.Append tDef
nWindEx.Close
```

Count Property Example (Microsoft Excel)

This example displays the number of recordsets in the Nwindex.mdb database and then enters their names on Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, i As Integer
Sheets("Sheet1").Activate
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Cells(1, 1).Value = "TableDef list for " & db.Name
Cells(1, 1).EntireColumn.AutoFit
For i = 0 To db.TableDefs.Count - 1
   Cells(i + 2, 1).Value = db.TableDefs(i).Name
Next i
MsgBox "There are " & db.TableDefs.Count & " TableDefs"
db.Close
```

DateCreated, LastUpdated Properties Example (Microsoft Excel)

This example displays the dates and times when the Customer recordset in the Nwindex.mdb database was created and last updated.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim creation As Variant, changed As Variant
Dim db As Database, td As TableDef
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set td = db.TableDefs("Customer")
creation = td.DateCreated
changed = td.LastUpdated
MsgBox "The " & td.Name & " table was created on " & creation _
   & " and updated on " & changed
db.Close
```

EditMode Property Example (Microsoft Excel)

This example checks to see whether the Customer recordset in the Nwindex.mdb database can be edited. If so, the example updates the value of the first field in the first record with the value in cell C3 on Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Customer")
If Not ((rs.EditMode = dbEditAdd) Or _
       (rs.EditMode = dbEditInProgress)) Then
   rs.Edit
   rs.Fields(0).Value = Worksheets(1).Cells(3, 3).Value
   rs.Update
Else
   MsgBox ("Cannot update database with cell value")
End If
rs.Close
db.Close
```

Filter Property Example (Microsoft Excel)

This example creates a new recordset that contains records from the Supplier recordset in the Nwindex.mdb database, and then it copies the recordset contents to Sheet1. These records contain only data on suppliers located in Canada. The example copies a new, sorted recordset to Microsoft Excel.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset, sortedSet As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Supplier", dbOpenDynaset)
rs.Filter = "[COUNTRY] = 'Canada'"
Set sortedSet = rs.OpenRecordset()
Sheets("Sheet1").Activate
ActiveCell.CopyFromRecordset sortedSet
sortedSet.Close
rs.Close
db.Close
```

LockEdits Property Example (Microsoft Excel)

This example locks the Customer recordset in the Nwindex.mdb database before updating the value in the CUSTMR_ID field of the first record with the value in cell A1 on Sheet1. Locking the recordset ensures that no other user can modify the record while it's being updated.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset
Sheets("Sheet1").Cells(1, 1).Value = "ACRIM"
databasePath = Application.Path & "\NWINDEX.MDB"
Set db = DBEngine.Workspaces(0).OpenDatabase(databasePath)
Set rs = db.OpenRecordset("Customer")
valueToAdd = Sheets("Sheet1").Cells(1, 1).Value
rs.LockEdits = False
rs.Edit
rs.fields("CUSTMR_ID").Value = valueToAdd
rs.Update
MsgBox "The new value of CUSTMR_ID is " & rs.fields("CUSTMR_ID").Value
rs.Close
db.Close
```

Name Property Example (Microsoft Excel)

This example enters in the active cell on Sheet1 the name of the first recordset in the Nwindex.mdb database.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, td As TableDef
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set td = db.TableDefs(0)
Sheets("Sheet1").Activate
ActiveCell.Value = td.Name
db.Close
```

NoMatch Property Example (Microsoft Excel)

This example adds all the names of contacts for the state of Washington to a list on worksheet one. The data is drawn from the Customer recordset in the Nwindex.mdb database.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database
Dim rs As Recordset
rw = 0
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("SELECT * FROM Customer")
criteria = "[REGION] = 'WA'"
Set wk = Worksheets.Add
rs.FindFirst criteria
Do Until rs.NoMatch
   rw = rw + 1
   wk.Range(rw, 1).Value = rs.fields("CONTACT").Value
   rs.FindNext criteria
Loop
rs.Close
db.Close
```

PercentPosition Property Example (Microsoft Excel)

This example selects and changes records in the Customer recordset in the Nwindex.mdb database, and then it copies the recordset to Sheet1. The status bar shows the percentage of records that have been changed.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset, sQLText as String
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
sQLText = "SELECT [CON_TITLE], [CONTACT], [COMPANY] FROM Customer " _
   & "WHERE [CON_TITLE] = 'Owner';"
Set rs = db.OpenRecordset(sQLText, dbOpenDynaset)
rs.MoveFirst
Sheets("Sheet1").Activate
Do While Not rs.EOF
   rs.Edit
   rs.Fields("CON_TITLE").Value = "Account Excecutive"
   rs.Update
   Application.StatusBar = rs.PercentPosition & "% of the" _
      & " records have been replaced."
   rs.MoveNext
Loop
Application.StatusBar = "Done"
rs.MoveFirst
numberOfRows = ActiveCell.CopyFromRecordset(rs)
MsgBox numberOfRows & " Records have been changed"
rs.Close
db.Close
```

RecordCount Property Example (Microsoft Excel)

This example displays the number of records in the Customer recordset in the Nwindex.mdb database.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path _
   & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Customer")
On Error GoTo errorHandler
rs.MoveLast
MsgBox "There are " & rs.RecordCount & " records in " _
   & rs.Name
rs.Close
db.Close
   Exit Sub
errorHandler:
   MsgBox "There are no records in " & rs.Name
   rs.Close
   db.Close
```

Size, Type Properties Example (Microsoft Excel)

This example copies to Sheet1 all fields of the **Double** type from Orddtail.dbf, a dBASE IV table located in the C:\Program Files\Common Files\Microsoft Shared\MSquery folder. (In Windows NT, Orddtail.dbf is located in the C:\Windows\Msapps\Msquery folder.)

```
Dim db As Database, recordsToCopy As Recordset, tDef As Recordset
Dim fieldsToStore(1000), fileName As String
fileName = "ORDDTAIL.DBF"
Set db = _
Workspaces(0).OpenDatabase("C:\Program Files\Common Files\Microsoft Shared\
MSquery", _
   False, False, "dBASE IV")
Set tDef = db.OpenRecordset(fileName)
n = 0
Sheets("Sheet1").Activate
For i = 0 To tDef.Fields.Count - 1
   If tDef.Fields(i).Type = dbDouble Then
      fieldsToStore(n) = tDef.fields(i).Name
      n = n + 1
   End If
Next
If fieldsToStore(0) = "" Then
   MsgBox "There are no number fields in this table."
   Exit Sub
End If
For i = 0 To n - 1
   records = "SELECT " & "[" & fieldsToStore(i) & "]" _
      & " from " & db.Recordsets(fileName).Name & ";"
   Set recordsToCopy = db.OpenRecordset(records)
   With ActiveSheet.Cells(1, i + 1)
      .CopyFromRecordset recordsToCopy
      .ColumnWidth = recordsToCopy.fields(0).Size
   End With
Next
recordsToCopy.Close
tDef.Close
db.Close
```

Sort Property Example (Microsoft Excel)

This example creates a new recordset from the Supplier recordset in the Nwindex.mdb database, and then it copies the new recordset to Sheet1. The new recordset is sorted on the COUNTRY field, in ascending order.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset, sortedSet As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Supplier", dbOpenDynaset)
rs.Sort = "[COUNTRY]"
Set sortedSet = rs.OpenRecordset()
Sheets("Sheet1").Activate
ActiveCell.CopyFromRecordset sortedSet
sortedSet.Close
rs.Close
db.Close
```

CreateQueryDef Method and SQL Property Example (Microsoft Excel)

This example creates a new query based on the Customer recordset in the Nwindex.mdb database. The query selects a snapshot of all customers in the state of Washington and then copies it to Sheet1.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, qDef As QueryDef, rs As Recordset
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set qDef = db.CreateQueryDef("WA Region")
qDef.SQL = "SELECT * FROM Customer WHERE [Region] = 'WA';"
Set rs = db.OpenRecordset("WA Region")
numberOfRows = Sheets("Sheet1").Cells(1, 1).CopyFromRecordset(rs)
Sheets("Sheet1").Activate
MsgBox numberOfRows & " records have been copied."
rs.Close
db.Close
```

Updatable Property Example (Microsoft Excel)

This example prompts the user to select a cell that contains a value for the CONTACT field of the Customer recordset in the Nwindex.mdb database. The example then checks to see whether the recordset can be updated. If so, the example adds a new record to the recordset, using the value in the selected cell.

To create the Nwindex.mdb database, run the Microsoft Excel example for the **CreateDatabase** method.

```
Dim db As Database, rs As Recordset
Sheets("Sheet1").Activate
cellToCopy = Application.InputBox("What cell value do you want" _
   & " to update as contact?", Type:=8)
If cellToCopy = False Then ' user cancelled InputBox
   Exit Sub
End If
Set db = Workspaces(0).OpenDatabase(Application.Path & "\NWINDEX.MDB")
Set rs = db.OpenRecordset("Customer")
If rs.Updatable = True Then
   rs.AddNew
   rs("CONTACT") = cellToCopy
   rs.Update
   rs.MoveLast
   MsgBox "The new contact is " & rs("CONTACT").Value
Else
   MsgBox "The recordset cannot be modified."
End If
rs.Close
db.Close
```

InputBox Function (Microsoft Excel)

In Microsoft Excel, the ***prompt*** string cannot contain more than 256 characters.

MsgBox Function (Microsoft Excel)

In Microsoft Excel, the ***prompt*** string cannot contain more than 256 characters.

Trappable Errors (Microsoft Excel)

Microsoft Excel uses the trappable errors listed in the following table, in addition to those used by Visual Basic. You can also work with Microsoft Excel cell error values in Visual Basic. For more information, see Cell Error Values.

Code	Message
1000	**'[Object]' does not have '[property name]' property** The property doesn't exist for this object. For more information, search Help for the object name.
1001	**'[Object]' does not have '[method name]' method** The method doesn't exist for this object. For more information, search Help for the object name.
1002	**Missing required argument '[argument]'** The method expected a required argument that wasn't specified. Add the argument to the code. To see a list of required arguments, search Help for the method name.
1003	**Invalid number of arguments** The method has the wrong number of arguments. This usually occurs when you specify arguments by position instead of by name and you have too many arguments. To see the a list of valid arguments for the method, search Help for the method name.
1004	**'[Method name]' method of '[object]' class failed** The method cannot be used on the object. Possible reasons for this include the following: • An argument contains a value that isn't valid. A common cause of this problem is an attempt to access an object that doesn't exist (for example, you tried to use `Workbooks(5)` when there were only three workbooks open). • The method cannot be used in the applied context. For example, some **Range** object methods require that the range contain data; if the range doesn't contain data, the method fails. • An external error occurred, such as a failure to read from or write to a file. For more information about the method, search Help for the method name.
1005	**Unable to set the '[property name]' property of the '[object]' class** The property cannot be changed. Possible reasons for this include the following: • The value you're using for the property isn't valid (for example, you set a property to a string value, but the property requires a Boolean value). • The property is read-only. For more information about the property, search Help for the property name.
1006	**Unable to get the '[property name]' property of the '[object]' class** The property cannot be read. A possible reason for this is that the

property cannot be used in the applied context. For example, the code `ActiveChart.Legend.Font.Color = RGB(255, 0, 0)` will cause this error if the active chart doesn't contain a legend.

For more information about the property, search Help for the property name.

Built-In Dialog Box Argument Lists

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscDialogArgListsC "}

Dialog box constant	Argument list(s)
xlDialogActivate	window_text, pane_num
xlDialogActiveCellFont	font, font_style, size, strikethrough, superscript, subscript, outline, shadow, underline, color, normal, background, start_char, char_count
xlDialogAddChartAutoformat	name_text, desc_text
xlDialogAddinManager	operation_num, addinname_text, copy_logical
xlDialogAlignment	horiz_align, wrap, vert_align, orientation, add_indent
xlDialogApplyNames	name_array, ignore, use_rowcol, omit_col, omit_row, order_num, append_last
xlDialogApplyStyle	style_text
xlDialogAppMove	x_num, y_num
xlDialogAppSize	x_num, y_num
xlDialogArrangeAll	arrange_num, active_doc, sync_horiz, sync_vert
xlDialogAssignToObject	macro_ref
xlDialogAssignToTool	bar_id, position, macro_ref
xlDialogAttachText	attach_to_num, series_num, point_num
xlDialogAttachToolbars	
xlDialogAutoCorrect	correct_initial_caps, capitalize_days
xlDialogAxes	x_primary, y_primary, x_secondary, y_secondary
xlDialogAxes	x_primary, y_primary, z_primary
xlDialogBorder	outline, left, right, top, bottom, shade, outline_color, left_color, right_color, top_color, bottom_color
xlDialogCalculation	type_num, iter, max_num, max_change, update, precision, date_1904, calc_save, save_values, alt_exp, alt_form
xlDialogCellProtection	locked, hidden
xlDialogChangeLink	old_text, new_text, type_of_link
xlDialogChartAddData	ref, rowcol, titles, categories, replace, series
xlDialogChartTrend	type, ord_per, forecast, backcast, intercept, equation, r_squared, name
xlDialogChartWizard	long, ref, gallery_num, type_num, plot_by, categories, ser_titles, legend, title, x_title, y_title, z_title, number_cats, number_titles
xlDialogCheckboxProperties	value, link, accel_text, accel2_text, 3d_shading
xlDialogClear	type_num
xlDialogColorPalette	file_text
xlDialogColumnWidth	width_num, reference, standard, type_num, standard_num
xlDialogCombination	type_num
xlDialogConsolidate	source_refs, function_num, top_row, left_col, create_links
xlDialogCopyChart	size_num
xlDialogCopyPicture	appearance_num, size_num, type_num
xlDialogCreateNames	top, left, bottom, right
xlDialogCreatePublisher	file_text, appearance, size, formats
xlDialogCustomizeToolbar	category
xlDialogDataDelete	
xlDialogDataLabel	show_option, auto_text, show_key
xlDialogDataSeries	rowcol, type_num, date_num, step_value, stop_value, trend
xlDialogDefineName	name_text, refers_to, macro_type, shortcut_text, hidden, category, local
xlDialogDefineStyle	style_text, number, font, alignment, border, pattern, protection
xlDialogDefineStyle	style_text, attribute_num, additional_def_args, ...
xlDialogDeleteFormat	format_text
xlDialogDeleteName	name_text
xlDialogDemote	row_col
xlDialogDisplay	formulas, gridlines, headings, zeros, color_num, reserved, outline, page_breaks, object_num

xlDialogDisplay	cell, formula, value, format, protection, names, precedents, dependents, note
xlDialogEditboxProperties	validation_num, multiline_logical, vscroll_logical, password_logical
xlDialogEditColor	color_num, red_value, green_value, blue_value
xlDialogEditDelete	shift_num
xlDialogEditionOptions	edition_type, edition_name, reference, option, appearance, size, formats
xlDialogEditSeries	series_num, name_ref, x_ref, y_ref, z_ref, plot_order
xlDialogErrorbarX	include, type, amount, minus
xlDialogErrorbarY	include, type, amount, minus
xlDialogExtract	unique
xlDialogFileDelete	file_text
xlDialogFillGroup	type_num
xlDialogFillWorkgroup	type_num
xlDialogFilterAdvanced	operation, list_ref, criteria_ref, copy_ref, unique
xlDialogFindFile	
xlDialogFont	name_text, size_num
xlDialogFontProperties	font, font_style, size, strikethrough, superscript, subscript, outline, shadow, underline, color, normal, background, start_char, char_count
xlDialogFormatAuto	format_num, number, font, alignment, border, pattern, width
xlDialogFormatChart	layer_num, view, overlap, angle, gap_width, gap_depth, chart_depth, doughnut_size, axis_num, drop, hilo, up_down, series_line, labels, vary
xlDialogFormatCharttype	apply_to, group_num, dimension, type_num
xlDialogFormatFont	color, backgd, apply, name_text, size_num, bold, italic, underline, strike, outline, shadow, object_id, start_num, char_num
xlDialogFormatFont	name_text, size_num, bold, italic, underline, strike, color, outline, shadow
xlDialogFormatFont	name_text, size_num, bold, italic, underline, strike, color, outline, shadow, object_id_text, start_num, char_num
xlDialogFormatLegend	position_num
xlDialogFormatMain	type_num, view, overlap, gap_width, vary, drop, hilo, angle, gap_depth, chart_depth, up_down, series_line, labels, doughnut_size
xlDialogFormatMove	x_offset, y_offset, reference
xlDialogFormatMove	x_pos, y_pos
xlDialogFormatMove	explosion_num
xlDialogFormatNumber	format_text
xlDialogFormatOverlay	type_num, view, overlap, gap_width, vary, drop, hilo, angle, series_dist, series_num, up_down, series_line, labels, doughnut_size
xlDialogFormatSize	width, height
xlDialogFormatSize	x_off, y_off, reference
xlDialogFormatText	x_align, y_align, orient_num, auto_text, auto_size, show_key, show_value, add_indent
xlDialogFormulaFind	text, in_num, at_num, by_num, dir_num, match_case, match_byte
xlDialogFormulaGoto	reference, corner
xlDialogFormulaReplace	find_text, replace_text, look_at, look_by, active_cell, match_case, match_byte
xlDialogFunctionWizard	
xlDialogGallery3dArea	type_num
xlDialogGallery3dBar	type_num
xlDialogGallery3dColumn	type_num
xlDialogGallery3dLine	type_num
xlDialogGallery3dPie	type_num
xlDialogGallery3dSurface	type_num
xlDialogGalleryArea	type_num, delete_overlay
xlDialogGalleryBar	type_num, delete_overlay
xlDialogGalleryColumn	type_num, delete_overlay
xlDialogGalleryCustom	name_text
xlDialogGalleryDoughnut	type_num, delete_overlay
xlDialogGalleryLine	type_num, delete_overlay
xlDialogGalleryPie	type_num, delete_overlay

xlDialogGalleryRadar	type_num, delete_overlay
xlDialogGalleryScatter	type_num, delete_overlay
xlDialogGoalSeek	target_cell, target_value, variable_cell
xlDialogGridlines	x_major, x_minor, y_major, y_minor, z_major, z_minor, 2D_effect
xlDialogInsert	shift_num
xlDialogInsertObject	object_class, file_name, link_logical, display_icon_logical, icon_file, icon_number, icon_label
xlDialogInsertPicture	file_name, filter_number
xlDialogInsertTitle	chart, y_primary, x_primary, y_secondary, x_secondary
xlDialogLabelProperties	accel_text, accel2_text, 3d_shading
xlDialogListboxProperties	range, link, drop_size, multi_select, 3d_shading
xlDialogMacroOptions	macro_name, description, menu_on, menu_text, shortcut_on, shortcut_key, function_category, status_bar_text, help_id, help_file
xlDialogMailEditMailer	to_recipients, cc_recipients, bcc_recipients, subject, enclosures, which_address
xlDialogMailLogon	name_text, password_text, download_logical
xlDialogMailNextLetter	
xlDialogMainChart	type_num, stack, 100, vary, overlap, drop, hilo, overlap%, cluster, angle
xlDialogMainChartType	type_num
xlDialogMenuEditor	
xlDialogMove	x_pos, y_pos, window_text
xlDialogNew	type_num, xy_series, add_logical
xlDialogNote	add_text, cell_ref, start_char, num_chars
xlDialogObjectProperties	placement_type, print_object
xlDialogObjectProtection	locked, lock_text
xlDialogOpen	file_text, update_links, read_only, format, prot_pwd, write_res_pwd, ignore_rorec, file_origin, custom_delimit, add_logical, editable, file_access, notify_logical, converter
xlDialogOpenLinks	document_text1, document_text2, ..., read_only, type_of_link
xlDialogOpenMail	subject, comments
xlDialogOpenText	file_name, file_origin, start_row, file_type, text_qualifier, consecutive_delim, tab, semicolon, comma, space, other, other_char, field_info
xlDialogOptionsCalculation	type_num, iter, max_num, max_change, update, precision, date_1904, calc_save, save_values
xlDialogOptionsChart	display_blanks, plot_visible, size_with_window
xlDialogOptionsEdit	incell_edit, drag_drop, alert, entermove, fixed, decimals, copy_objects, update_links, move_direction, autocomplete, animations
xlDialogOptionsGeneral	R1C1_mode, dde_on, sum_info, tips, recent_files, old_menus, user_info, font_name, font_size, default_location, alternate_location, sheet_num, enable_under
xlDialogOptionsListsAdd	string_array
xlDialogOptionsListsAdd	import_ref, by_row
xlDialogOptionsTransition	menu_key, menu_key_action, nav_keys, trans_eval, trans_entry
xlDialogOptionsView	formula, status, notes, show_info, object_num, page_breaks, formulas, gridlines, color_num, headers, outline, zeros, hor_scroll, vert_scroll, sheet_tabs
xlDialogOutline	auto_styles, row_dir, col_dir, create_apply
xlDialogOverlay	type_num, stack, 100, vary, overlap, drop, hilo, overlap%, cluster, angle, series_num, auto
xlDialogOverlayChartType	type_num
xlDialogPageSetup	head, foot, left, right, top, bot, hdng, grid, h_cntr, v_cntr, orient, paper_size, scale, pg_num, pg_order, bw_cells, quality, head_margin, foot_margin, notes, draft
xlDialogPageSetup	head, foot, left, right, top, bot, size, h_cntr, v_cntr, orient, paper_size, scale, pg_num, bw_chart, quality, head_margin, foot_margin, draft
xlDialogPageSetup	head, foot, left, right, top, bot, orient, paper_size, scale, quality, head_margin, foot_margin, pg_num
xlDialogParse	parse_text, destination_ref
xlDialogPasteSpecial	paste_num, operation_num, skip_blanks, transpose

xlDialogPasteSpecial	rowcol, titles, categories, replace, series
xlDialogPasteSpecial	paste_num
xlDialogPasteSpecial	format_text, pastelink_logical, display_icon_logical, icon_file, icon_number, icon_label
xlDialogPatterns	apattern, afore, aback, newui
xlDialogPatterns	lauto, lstyle, lcolor, lwt, hwidth, hlength, htype
xlDialogPatterns	bauto, bstyle, bcolor, bwt, shadow, aauto, apattern, afore, aback, rounded, newui
xlDialogPatterns	bauto, bstyle, bcolor, bwt, shadow, aauto, apattern, afore, aback, invert, apply, newfill
xlDialogPatterns	lauto, lstyle, lcolor, lwt, tmajor, tminor, tlabel
xlDialogPatterns	lauto, lstyle, lcolor, lwt, apply, smooth
xlDialogPatterns	lauto, lstyle, lcolor, lwt, mauto, mstyle, mfore, mback, apply, smooth
xlDialogPatterns	type, picture_units, apply
xlDialogPivotFieldGroup	start, end, by, periods
xlDialogPivotFieldProperties	name, pivot_field_name, new_name, orientation, function, formats
xlDialogPivotFieldUngroup	
xlDialogPivotShowPages	name, page_field
xlDialogPivotTableWizard	type, source, destination, name, row_grand, col_grand, save_data, apply_auto_format, auto_page, reserved
xlDialogPlacement	placement_type
xlDialogPrint	range_num, from, to, copies, draft, preview, print_what, color, feed, quality, y_resolution, selection, printer_text, print_to_file, collate
xlDialogPrinterSetup	printer_text
xlDialogPrintPreview	
xlDialogPromote	rowcol
xlDialogProperties	title, subject, author, keywords, comments
xlDialogProtectDocument	contents, windows, password, objects, scenarios
xlDialogPushbuttonProperties	default_logical, cancel_logical, dismiss_logical, help_logical, accel_text, accel_text2
xlDialogReplaceFont	font_num, name_text, size_num, bold, italic, underline, strike, color, outline, shadow
xlDialogRoutingSlip	recipients, subject, message, route_num, return_logical, status_logical
xlDialogRowHeight	height_num, reference, standard_height, type_num
xlDialogRun	reference, step
xlDialogSaveAs	document_text, type_num, prot_pwd, backup, write_res_pwd, read_only_rec
xlDialogSaveCopyAs	document_text
xlDialogSaveNewObject	
xlDialogSaveWorkbook	document_text, type_num, prot_pwd, backup, write_res_pwd, read_only_rec
xlDialogSaveWorkspace	name_text
xlDialogScale	cross, cat_labels, cat_marks, between, max, reverse
xlDialogScale	min_num, max_num, major, minor, cross, logarithmic, reverse, max
xlDialogScale	cat_labels, cat_marks, reverse, between
xlDialogScale	series_labels, series_marks, reverse
xlDialogScale	min_num, max_num, major, minor, cross, logarithmic, reverse, min
xlDialogScenarioAdd	scen_name, value_array, changing_ref, scen_comment, locked, hidden
xlDialogScenarioCells	changing_ref
xlDialogScenarioEdit	scen_name, new_scenname, value_array, changing_ref, scen_comment, locked, hidden
xlDialogScenarioMerge	source_file
xlDialogScenarioSummary	result_ref, report_type
xlDialogScrollbarProperties	value, min, max, inc, page, link, 3d_shading
xlDialogSelectSpecial	type_num, value_type, levels
xlDialogSendMail	recipients, subject, return_receipt
xlDialogSeriesAxes	axis_num
xlDialogSeriesOrder	chart_num, old_series_num, new_series_num

xlDialogSeriesX	x_ref
xlDialogSeriesY	name_ref, y_ref
xlDialogSetControlValue	value
xlDialogSetPrintTitles	titles_for_cols_ref, titles_for_rows_ref
xlDialogSetUpdateStatus	link_text, status, type_of_link
xlDialogShowDetail	rowcol, rowcol_num, expand, show_field
xlDialogShowToolbar	bar_id, visible, dock, x_pos, y_pos, width, protect, tool_tips, large_buttons, color_buttons
xlDialogSize	width, height, window_text
xlDialogSort	orientation, key1, order1, key2, order2, key3, order3, header, custom, case
xlDialogSort	orientation, key1, order1, type, custom
xlDialogSortSpecial	sort_by, method, key1, order1, key2, order2, key3, order3, header, order, case
xlDialogSplit	col_split, row_split
xlDialogStandardFont	name_text, size_num, bold, italic, underline, strike, color, outline, shadow
xlDialogStandardWidth	standard_num
xlDialogStyle	bold, italic
xlDialogSubscribeTo	file_text, format_num
xlDialogSubtotalCreate	at_change_in, function_num, total, replace, pagebreaks, summary_below
xlDialogSummaryInfo	title, subject, author, keywords, comments
xlDialogTable	row_ref, column_ref
xlDialogTabOrder	
xlDialogTextToColumns	destination_ref, data_type, text_delim, consecutive_delim, tab, semicolon, comma, space, other, other_char, field_info
xlDialogUnhide	window_text
xlDialogUpdateLink	link_text, type_of_link
xlDialogVbaInsertFile	filename_text
xlDialogVbaMakeAddin	filename_text
xlDialogVbaProcedureDefinition	
xlDialogView3d	elevation, perspective, rotation, axes, height%, autoscale
xlDialogWindowMove	x_pos, y_pos, window_text
xlDialogWindowSize	width, height, window_text
xlDialogWorkbookAdd	name_array, dest_book, position_num
xlDialogWorkbookCopy	name_array, dest_book, position_num
xlDialogWorkbookInsert	type_num
xlDialogWorkbookMove	name_array, dest_book, position_num
xlDialogWorkbookName	oldname_text, newname_text
xlDialogWorkbookNew	
xlDialogWorkbookOptions	sheet_name, bound_logical, new_name
xlDialogWorkbookProtect	structure, windows, password
xlDialogWorkbookTabSplit	ratio_num
xlDialogWorkbookUnhide	sheet_text
xlDialogWorkgroup	name_array
xlDialogWorkspace	fixed, decimals, r1c1, scroll, status, formula, menu_key, remote, entermove, underlines, tools, notes, nav_keys, menu_key_action, drag_drop, show_info
xlDialogZoom	magnification

LinkSource Property (Microsoft Excel)

For Microsoft Excel, the source is a name in the workbook.

Add Method (Microsoft Excel)

For Microsoft Excel, the ***linkSource*** argument specifies a name in the workbook.

Add Method Example (Microsoft Excel)

This example adds a static custom property named "Complete."

```
With ActiveWorkbook.CustomDocumentProperties
   .Add name := "Complete", linkToContent := False, _
      type := offPropertyTypeBoolean, value := False
End With
```

This example adds a property linked to the name "Grand_Total."

```
With ActiveWorkbook.CustomDocumentProperties
   .Add name := "Total Sales", linkToContent := True, _
      linkSource := "Grand_Total"
End With
```

Count Property Example (Microsoft Excel)

This example displays the number of custom document properties in the active workbook.

```
MsgBox ActiveWorkbook.CustomDocumentProperties.Count
```

Delete Method Example (Microsoft Excel)

This example deletes the "Total Sales" custom document property from the active workbook.

```
ActiveWorkbook.CustomDocumentProperties("Total Sales").Delete
```

Item Method Example (Microsoft Excel)

This example creates a list that contains the names of all the built-in document properties in the active workbook.

```
Set wk = Worksheets(1)
Set builtinProps = ActiveWorkbook.BuiltinDocumentProperties
For i = 1 To builtinProps.Count
   wk.Cells(i, 1).Value = builtinProps.Item(i).Name
Next
```

LinkToContent Property Example (Microsoft Excel)

This example displays the linked status for each custom document property in the active workbook as a list on worksheet one.

```
rw = 1
With Worksheets(1)
   For Each pro In ActiveWorkbook.CustomDocumentProperties
      .Cells(rw, 1) = pro.Name
      .Cells(rw, 2) = pro.LinkToContent
      If pro.LinkToContent Then
         .Cells(rw, 3) = pro.LinkSource
      End If
      rw = rw + 1
   Next
End With
```

LinkSource Property Example (Microsoft Excel)

This example links the custom document property named "TotalSales" to the named range "SalesNumbers" in the active workbook.

```
ActiveWorkbook.CustomDocumentProperties _
    .Item("TotalSales").LinkSource = "SalesNumbers"
```

Name Property Example (Microsoft Excel)

This example displays the name of custom document property one in the active workbook.

```
MsgBox ActiveWorkbook.CustomDocumentProperties(1).Name
```

Type Property Example (Microsoft Excel)

This example displays the type of custom document property one in the active workbook.

```
MsgBox ActiveWorkbook.CustomDocumentProperties(1).Type
```

Value Property Example (Microsoft Excel)

This example displays the value of the first custom document property in the active workbook.

```
MsgBox ActiveWorkbook.CustomDocumentProperties(1).Value
```

Array Function Example (Microsoft Excel)

This example fills the range A1:C5 on Sheet1, Sheet5, and Sheet7 with the contents of the same range on Sheet1.

```
x = Array("Sheet1", "Sheet5", "Sheet7")
Sheets(x).FillAcrossSheets _
    Worksheets("Sheet1").Range("A1:C5")
```

This example consolidates data from Sheet2 and Sheet3 onto Sheet1, using the SUM function.

```
Worksheets("Sheet1").Range("A1").Consolidate _
    sources:=Array("Sheet2!R1C1:R37C6", "Sheet3!R1C1:R37C6"), _
    Function:=xlSum
```

This example adds an array of strings as a custom list.

```
Application.AddCustomList Array("cogs", "sprockets", _
    "widgets", "gizmos")
```

This example hides Chart1, Chart3, and Chart5. Note that in this example, the **Charts** property returns a **Sheets** object instead of a **Charts** object.

```
Charts(Array("Chart1", "Chart3", "Chart5")).Visible = False
```

This example sets the entries in list box one on Dialog1.

```
DialogSheets("Dialog1").ListBoxes(1).List = _
    Array("cogs", "widgets", "sprockets", "gizmos")
```

This example creates a group that includes drawing objects one, three, and five on Sheet1.

```
Set myGroup = Worksheets("Sheet1").DrawingObjects(Array(1, 3, 5)).Group
Worksheets("Sheet1").Activate
myGroup.Select
```

Chr Function Example (Microsoft Excel)

This example fills list box one on Dialog1 with the letters A through Z.

```
For i = 65 To 90
   DialogSheets("Dialog1").ListBoxes(1).AddItem Text:=Chr(i)
Next i
```

This example creates a line break in a message box by using the **Chr** function.

```
msgText = "The current folder is:" & Chr(13) & CurDir()
MsgBox msgText
```

CVErr Function Example (Microsoft Excel)

This example inserts the seven cell error values into cells A1:A7 on Sheet1.

```
myArray = Array(xlErrDiv0, xlErrNA, xlErrName, xlErrNull, _
   xlErrNum, xlErrRef, xlErrValue)
For i = 1 To 7
   Worksheets("Sheet1").Cells(i, 1).Value = CVErr(myArray(i - 1))
Next i
```

This example displays a message if the active cell on Sheet1 contains a cell error value. You can use this example as a framework for a cell-error-value error handler.

```
Worksheets("Sheet1").Activate
If IsError(ActiveCell.Value) Then
   errval = ActiveCell.Value
   Select Case errval
      Case CVErr(xlErrDiv0)
         MsgBox "#DIV/0! error"
      Case CVErr(xlErrNA)
         MsgBox "#N/A error"
      Case CVErr(xlErrName)
         MsgBox "#NAME? error"
      Case CVErr(xlErrNull)
         MsgBox "#NULL! error"
      Case CVErr(xlErrNum)
         MsgBox "#NUM! error"
      Case CVErr(xlErrRef)
         MsgBox "#REF! error"
      Case CVErr(xlErrValue)
         MsgBox "#VALUE! error"
      Case Else
         MsgBox "This should never happen!!"
   End Select
End If
```

Do...Loop Statement Example (Microsoft Excel)

This example sorts the data in the first column on Sheet1 and then deletes any rows that contain duplicate data.

```
Worksheets("Sheet1").Range("A1").Sort _
   key1:=Worksheets("Sheet1").Range("A1")
Set currentCell = Worksheets("Sheet1").Range("A1")
Do While Not IsEmpty(currentCell)
   Set nextCell = currentCell.Offset(1, 0)
   If nextCell.Value = currentCell.Value Then
      currentCell.EntireRow.Delete
   End If
   Set currentCell = nextCell
Loop
```

For Each...Next Statement Example (Microsoft Excel)

This example loops on cells A1:D10 on Sheet1. If one of the cells has a value less than 0.001, the code replaces the value with 0 (zero).

```
For Each c in Worksheets("Sheet1").Range("A1:D10")
   If c.Value < .001 Then
      c.Value = 0
   End If
Next c
```

This example loops on the range named "TestRange" and then displays the number of empty cells in the range.

```
numBlanks = 0
For Each c In Range("TestRange")
   If c.Value = "" Then
      numBlanks = numBlanks + 1
   End If
Next c
MsgBox "There are " & numBlanks & " empty cells in this range."
```

This example closes and saves changes to all workbooks except the one that's running the example.

```
For Each w In Workbooks
   If w.Name <> ThisWorkbook.Name Then
      w.Close savechanges:=True
   End If
Next w
```

This example deletes every worksheet in the active workbook without displaying the confirmation dialog box. There must be at least one other visible sheet in the workbook.

```
Application.DisplayAlerts = False
For Each w In Worksheets
   w.Delete
Next w
Application.DisplayAlerts = True
```

This example creates a new worksheet and then inserts into it a list of all the names in the active workbook, including their formulas in A1-style notation in the language of the user.

```
Set newSheet = ActiveWorkbook.Worksheets.Add
i = 1
For Each nm In ActiveWorkbook.Names
   newSheet.Cells(i, 1).Value = nm.NameLocal
   newSheet.Cells(i, 2).Value = "'" & nm.RefersToLocal
   i = i + 1
Next nm
```

For...Next Statement Example (Microsoft Excel)

This example displays the number of columns in the selection on Sheet1. The code also tests for a multiple-area selection; if one exists, the code loops on the areas of the selection.

```
Worksheets("Sheet1").Activate
areaCount = Selection.Areas.Count
If areaCount <= 1 Then
   MsgBox "The selection contains " & _
      Selection.Columns.Count & " columns."
Else
   For i = 1 To areaCount
      MsgBox "Area " & i & " of the selection contains " & _
         Selection.Areas(i).Columns.Count & " columns."
   Next i
End If
```

This example creates a new worksheet and then inserts a list of the active workbook's sheet names into the first column of the worksheet.

```
Set newSheet = Sheets.Add(Type:=xlWorksheet)
For i = 1 To Sheets.Count
   newSheet.Cells(i, 1).Value = Sheets(i).Name
Next i
```

This example selects every other item in list box one on Sheet1.

```
Dim items() As Boolean
Set lbox = Worksheets("Sheet1").ListBoxes(1)
ReDim items(1 To lbox.ListCount)
For i = 1 To lbox.ListCount
   If i Mod 2 = 1 Then
      items(i) = True
   Else
      items(i) = False
   End If
Next
lbox.MultiSelect = xlExtended
lbox.Selected = items
```

If...Then...Else Statement Example (Microsoft Excel)

This example loops on cells A1:D10 on Sheet1. If one of the cells has a value less than 0.001, the code replaces the value with 0 (zero).

```
For Each c in Worksheets("Sheet1").Range("A1:D10")
   If c.Value < .001 Then
      c.Value = 0
   End If
Next c
```

This example loops on the range named "TestRange" and then displays the number of empty cells in the range.

```
numBlanks = 0
For Each c In Range("TestRange")
   If c.Value = "" Then
      numBlanks = numBlanks + 1
   End If
Next c
MsgBox "There are " & numBlanks & " empty cells in this range."
```

This example sets the standard font to Geneva (on the Macintosh) or Arial (in Windows).

```
If Application.OperatingSystem Like "*Macintosh*" Then
   Application.StandardFont = "Geneva"
Else
   Application.StandardFont = "Arial"
End If
```

Is Operator Example (Microsoft Excel)

This example selects the intersection of two named ranges ("rg1" and "rg2") on Sheet1. If the ranges don't intersect, the example displays a message.

```
Worksheets("Sheet1").Activate
Set isect = Application.Intersect(Range("rg1"), Range("rg2"))
If isect Is Nothing Then
   MsgBox "Ranges do not intersect"
Else
   isect.Select
End If
```

This example finds the first occurrence of the word "Phoenix" in column B on Sheet1 and then displays the address of the cell that contains this word. If the word isn't found, the example diplays a message.

```
Set foundCell = Worksheets("Sheet1").Columns("B").Find("Phoenix")
If foundCell Is Nothing Then
   MsgBox "The word was not found"
Else
   MsgBox "The word was found in cell " & foundCell.Address
End If
```

IsEmpty Function Example (Microsoft Excel)

This example sorts the data in the first column on Sheet1 and then deletes any rows that contain duplicate data.

```
Worksheets("Sheet1").Range("A1").Sort _
   key1:=Worksheets("Sheet1").Range("A1")
Set currentCell = Worksheets("Sheet1").Range("A1")
Do While Not IsEmpty(currentCell)
   Set nextCell = currentCell.Offset(1, 0)
   If nextCell.Value = currentCell.Value Then
      currentCell.EntireRow.Delete
   End If
   Set currentCell = nextCell
Loop
```

IsNull Function Example (Microsoft Excel)

This example creates a list of registered functions, placing each one in a separate row on Sheet1. Column A contains the full path and file name of the DLL or code resource, column B contains the function name, and column C contains the code for the argument data type.

```
theArray = Application.RegisteredFunctions
If IsNull(theArray) Then
   MsgBox "No registered functions"
Else
   For i = LBound(theArray) To UBound(theArray)
      For j = 1 To 3
         Worksheets("Sheet1").Cells(i, j).Formula = theArray(i, j)
      Next j
   Next i
End If
```

LBound Function Example (Microsoft Excel)

This example writes the elements of the first custom list in column one on Sheet1.

```
listArray = Application.GetCustomListContents(1)
For i = LBound(listArray, 1) To UBound(listArray, 1)
   Worksheets("sheet1").Cells(i, 1).Value = listArray(i)
Next i
```

This example assumes that you used an external data source to create a PivotTable on Sheet1 in the active workbook. The example inserts the SQL connection string and query string into a new worksheet.

```
Set newSheet = ActiveWorkbook.Worksheets.Add
sdArray = Worksheets("Sheet1").UsedRange.PivotTable.SourceData
For i = LBound(sdArray) To UBound(sdArray)
   newSheet.Cells(i, 1) = sdArray(i)
Next i
```

Like Operator Example (Microsoft Excel)

This example deletes every defined name that contains "temp". The `Option Compare Text` statement must be included at the top of any module that contains this example.

```
For Each nm In ActiveWorkbook.Names
   If nm.Name Like "*temp*" Then
      nm.Delete
   End If
Next nm
```

This example adds an arrowhead to every shape on Sheet1 that has the word "Line" in its name.

```
For Each d In Worksheets("Sheet1").DrawingObjects
   If d.Name Like "*Line*" Then
      d.ArrowHeadLength = xlLong
      d.ArrowHeadStyle = xlOpen
      d.ArrowHeadWidth = xlNarrow
   End If
Next
```

Mod Operator Example (Microsoft Excel)

This example sets the column width of every other column on Sheet1 to 4 points.

```
For Each col In Worksheets("Sheet1").Columns
   If col.Column Mod 2 = 0 Then
      col.ColumnWidth = 4
   End If
Next col
```

This example sets the row height of every other row on Sheet1 to 4 points.

```
For Each rw In Worksheets("Sheet1").Rows
   If rw.Row Mod 2 = 0 Then
      rw.RowHeight = 4
   End If
Next rw
```

This example selects every other item in list box one on Sheet1.

```
Dim items() As Boolean
Set lbox = Worksheets("Sheet1").ListBoxes(1)
ReDim items(1 To lbox.ListCount)
For i = 1 To lbox.ListCount
   If i Mod 2 = 1 Then
      items(i) = True
   Else
      items(i) = False
   End If
Next
lbox.MultiSelect = xlExtended
lbox.Selected = items
```

RGB Function Example (Microsoft Excel)

This example sets the gridline color in the active window in Book1.xls to red.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.GridlineColor = RGB(255,0,0)
```

This example sets the color of the tick labels on the value axis in Chart1 to green.

```
Charts("Chart1").Axes(xlValue).TickLabels.Font.Color = RGB(0, 255, 0)
```

This example sets the marker background and foreground colors for the second point in series one in Chart1 to green and red, respectively.

```
With Charts("Chart1").SeriesCollection(1).Points(2)
    .MarkerBackgroundColor = RGB(0,255,0) ' green
    .MarkerForegroundColor = RGB(255,0,0) ' red
End With
```

This example sets the interior pattern color for rectangle one on Sheet1 to red.

```
With Worksheets("Sheet1").Rectangles(1).Interior
    .Pattern = xlGrid
    .PatternColor = RGB(255,0,0)
End With
```

Select Case Statement Example (Microsoft Excel)

This example displays the name of the mail system installed on the computer.

```
Select Case Application.MailSystem
   Case Is = xlMAPI
      MsgBox "Mail system is Microsoft Mail"
   Case Is = xlPowerTalk
      MsgBox "Mail system is PowerTalk"
   Case Is = xlNoMailSystem
      MsgBox "No mail system installed"
End Select
```

This example displays a message box that indicates the location of the active cell in the PivotTable.

```
Worksheets("Sheet1").Activate
Select Case ActiveCell.LocationInTable
Case Is = xlRowHeader
   MsgBox "Active cell is part of a row header"
Case Is = xlColumnHeader
   MsgBox "Active cell is part of a column header"
Case Is = xlPageHeader
   MsgBox "Active cell is part of a page header"
Case Is = xlDataHeader
   MsgBox "Active cell is part of a data header"
Case Is = xlRowItem
   MsgBox "Active cell is part of a row item"
Case Is = xlColumnItem
   MsgBox "Active cell is part of a column item"
Case Is = xlPageItem
   MsgBox "Active cell is part of a page item"
Case Is = xlDataItem
   MsgBox "Active cell is part of a data item"
Case Is = xlTableBody
   MsgBox "Active cell is part of the table body"
End Select
```

This example displays a message if the active cell on Sheet1 contains a cell error value. You can use this example as a framework for a cell-error-value error handler.

```
Worksheets("Sheet1").Activate
If IsError(ActiveCell.Value) Then
   errval = ActiveCell.Value
   Select Case errval
      Case CVErr(xlErrDiv0)
         MsgBox "#DIV/0! error"
      Case CVErr(xlErrNA)
         MsgBox "#N/A error"
      Case CVErr(xlErrName)
         MsgBox "#NAME? error"
      Case CVErr(xlErrNull)
         MsgBox "#NULL! error"
      Case CVErr(xlErrNum)
         MsgBox "#NUM! error"
      Case CVErr(xlErrRef)
         MsgBox "#REF! error"
      Case CVErr(xlErrValue)
         MsgBox "#VALUE! error"
```

```
      Case Else
         MsgBox "This should never happen!!"
   End Select
End If
```

Set Statement Example (Microsoft Excel)

This example adds a new worksheet to the active workbook and then sets the name of the worksheet.

```
Set newSheet = Worksheets.Add
newSheet.Name = "1995 Budget"
```

This example creates a new worksheet and then inserts into it a list of all the names in the active workbook, including their formulas in A1-style notation in the language of the user.

```
Set newSheet = ActiveWorkbook.Worksheets.Add
i = 1
For Each nm In ActiveWorkbook.Names
   newSheet.Cells(i, 1).Value = nm.NameLocal
   newSheet.Cells(i, 2).Value = "'" & nm.RefersToLocal
   i = i + 1
Next
```

Static Statement Example (Microsoft Excel)

This example uses the worksheet function **Pmt** to calculate a home mortgage loan payment. Note that this example uses the **InputBox** method instead of the **InputBox** function so that the method can perform type checking. The **Static** statements cause Visual Basic to retain the values of the three variables; these are displayed as default values the next time you run the example.

```
Static loanAmt
Static loanInt
Static loanTerm
loanAmt = Application.InputBox _
   (Prompt:="Loan amount (100,000 for example)", _
      Default:=loanAmt, Type:=1)
loanInt = Application.InputBox _
   (Prompt:="Annual interest rate (8.75 for example)", _
      Default:=loanInt, Type:=1)
loanTerm = Application.InputBox _
   (Prompt:="Term in years (30 for example)", _
      Default:=loanTerm, Type:=1)
payment = Application.Pmt(loanInt / 1200, loanTerm * 12, loanAmt)
MsgBox "Monthly payment is " & Format(payment, "Currency")
```

TypeName Function Example (Microsoft Excel)

This example displays the Visual Basic object type of the selection. You can run this example with cells selected, with a single oval selected, or with several different graphic objects selected.

```
Worksheets("Sheet1").Activate
MsgBox "The selection object type is " & TypeName(Selection)
```

UBound Function Example (Microsoft Excel)

This example writes the elements of the first custom list in column one on Sheet1.

```
listArray = Application.GetCustomListContents(1)
For i = LBound(listArray, 1) To UBound(listArray, 1)
   Worksheets("sheet1").Cells(i, 1).Value = listArray(i)
Next i
```

This example assumes that you used an external data source to create a PivotTable on Sheet1. The example inserts the SQL connection string and query string into a new worksheet.

```
Set newSheet = ActiveWorkbook.Worksheets.Add
sdArray = Worksheets("Sheet1").UsedRange.PivotTable.SourceData
For i = LBound(sdArray) To UBound(sdArray)
   newSheet.Cells(i, 1) = sdArray(i)
Next i
```

With Statement Example (Microsoft Excel)

This example creates a formatted multiplication table in cells A1:K11 on Sheet1.

```
Set dataTableRange = Worksheets("Sheet1").Range("A1:K11")
Set rowInputCell = Worksheets("Sheet1").Range("A12")
Set columnInputCell = Worksheets("Sheet1").Range("A13")

Worksheets("Sheet1").Range("A1").Formula = "=A12*A13"
For i = 2 To 11
   Worksheets("Sheet1").Cells(i, 1) = i - 1
   Worksheets("Sheet1").Cells(1, i) = i - 1
Next i
dataTableRange.Table rowInputCell, columnInputCell
With Worksheets("Sheet1").Range("A1").CurrentRegion
   .Rows(1).Font.Bold = True
   .Columns(1).Font.Bold = True
   .Columns.AutoFit
End With
```

Add Method (AddIns Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddAddInsObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddAddInsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddAddInsObjA "}

Adds a new add-in file to the list of add-ins. Returns an **AddIn** object.

Syntax

expression.**Add(*Filename, CopyFile*)**

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

Filename Required **String**. The name of the file that contains the add-in you want to add to the list in the add-in manager.

CopyFile Optional **Variant**. Ignored if the add-in file is on a hard disk. **True** to copy the add-in to your hard disk, if the add-in is on a removable medium (a floppy disk or compact disc). **False** to have the add-in remain on the removable medium. If this argument is omitted, Microsoft Excel displays a dialog box and asks you to choose.

Remarks

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

Add Method (AddIns Collection) Example

This example inserts the add-in Myaddin.xla from drive A. When you run this example, Microsoft Excel copies the file A:\Myaddin.xla to the Library folder on your hard disk and adds the add-in title to the list in the **Add-Ins** dialog box.

```
Set myAddIn = AddIns.Add(Filename:="A:\MYADDIN.XLA", _
      CopyFile:=True)
MsgBox myAddIn.Title & " has been added to the list"
```

On the Macintosh, this example copies the add-in from the disk labeled "Add-In Disk" to the Macro Library folder.

```
Set myAddIn = AddIns.Add(Filename:="Add-In Disk:My Add-In", _
      CopyFile:=True)
MsgBox myAddIn.Title & " has been added to the list"
```

Add Method (Charts Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddChartsObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddChartsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddChartsObjA "}

Creates a new chart sheet. Returns a **Chart** object.

Syntax

expression.**Add(*Before, After, Count*)**

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

Before Optional **Variant**. An object that specifies the sheet before which the new sheet is added.

After Optional **Variant**. An object that specifies the sheet after which the new sheet is added.

Count Optional **Variant**. The number of sheets to be added. The default value is one.

Remarks

If ***Before*** and ***After*** are both omitted, the new chart is inserted before the active sheet.

Add Method (Charts Collection) Example

This example creates an empty chart sheet and inserts it before the last worksheet.

```
ActiveWorkbook.Charts.Add Before:=Worksheets(Worksheets.Count)
```

Add Method (Names Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddNamesObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddNamesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddNamesObjA "}

Defines a new name. Returns a **Name** object.

Syntax

expression.**Add(*Name, RefersTo, Visible, MacroType, ShortcutKey, Category, NameLocal, RefersToLocal, CategoryLocal, RefersToR1C1, RefersToR1C1Local*)**

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

Name Optional **Variant**. Required if ***NameLocal*** isn't specified. The text to use as the name (in the language of the macro). Names cannot include spaces and cannot look like cell references.

RefersTo Optional **Variant**. Required unless one of the other ***RefersTo*** arguments is specified. Describes what the name refers to (in the language of the macro, using A1-style notation).

Visible Optional **Variant**. **True** to define the name normally. **False** to define the name as a hidden name (that is, it doesn't appear in either the **Define Name**, **Paste Name**, or **Goto** dialog box). The default value is **True**.

MacroType Optional **Variant**. The macro type, as shown in the following table.

Value	Meaning
1	User-defined function (**Function** procedure)
2	Macro (also known as **Sub** procedure)
3 or omitted	None (that is, the name doesn't refer to a user-defined function or macro)

ShortcutKey Optional **Variant**. The macro shortcut key. Must be a single letter, such as "z" or "Z". Applies only for command macros.

Category Optional **Variant**. The category of the macro or function if ***MacroType*** is 1 or 2. The category is used in the Function Wizard. Existing categories can be referred to either by number (starting at 1) or by name (in the language of the macro). Microsoft Excel creates a new category if the specified category doesn't already exist.

NameLocal Optional **Variant**. Required if ***Name*** isn't specified. The text to use as the name (in the language of the user). Names cannot include spaces and cannot look like cell references.

RefersToLocal Optional **Variant**. Required unless one of the other ***RefersTo*** arguments is specified. Describes what the name refers to (in the language of the user, using A1-style notation).

CategoryLocal Optional **Variant**. Required if ***Category*** isn't specified. Text identifying the category of a custom function in the language of the user.

RefersToR1C1 Optional **Variant**. Required unless one of the other ***RefersTo*** arguments is specified. Describes what the name refers to (in the language of the macro, using R1C1-style notation).

RefersToR1C1Local Optional **Variant**. Required unless one of the other ***RefersTo*** arguments is specified. Describes what the name refers to (in the language of the user, using R1C1-style notation).

Add Method (Names Collection) Example

This example defines a new name for the range A1:D3 on Sheet1 in the active workbook.

```
ActiveWorkbook.Names.Add _
   Name:="tempRange", _
   RefersTo:="=Sheet1!$A$1:$D$3"
```

Add Method (Scenarios Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddScenariosObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddScenariosObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddScenariosObjA "}

Creates a new scenario and adds it to the list of scenarios that are available for the current worksheet. Returns a **Scenario** object.

Syntax

expression.**Add(*Name, ChangingCells, Values, Comment, Locked, Hidden*)**

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

Name Required **String**. The scenario name.

ChangingCells Required **Variant**. A **Range** object that refers to the changing cells for the scenario.

Values Optional **Variant**. An array that contains the scenario values for the cells in ***ChangingCells***. If this argument is omitted, the scenario values are assumed to be the current values in the cells in***ChangingCells***.

Comment Optional **Variant**. A string that specifies comment text for the scenario. When a new scenario is added, the author name and date are automatically added at the beginning of the comment text.

Locked Optional **Variant**. **True** to lock the scenario to prevent changes. The default value is **True**.

Hidden Optional **Variant**. **True** to hide the scenario. The default value is **False**.

Remarks

A scenario name must be unique; Microsoft Excel generates an error if you try to create a scenario with a name that's already in use.

Add Method (Scenarios Collection) Example

This example adds a new scenario to Sheet1.

```
Worksheets("Sheet1").Scenarios.Add Name:="Best Case", _
   ChangingCells:=Worksheets("Sheet1").Range("A1:A4"), _
   Values:=Array(23, 5, 6, 21), _
   Comment:="Most favorable outcome."
```

Add Method (Sheets Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddSheetsObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddSheetsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddSheetsObjA "}

Creates a new worksheet, chart, or macro sheet.

Syntax

expression.**Add(*Before, After, Count, Type*)**

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

Before Optional **Variant**. An object that specifies the sheet before which the new sheet is added.

After Optional **Variant**. An object that specifies the sheet after which the new sheet is added.

Count Optional **Variant**. The number of sheets to be added. The default value is one.

Type Optional **Variant**. Specifies the sheet type. Can be one of the following **XlSheetType** constants: **xlWorksheet**, **xlChart**, **xlExcel4MacroSheet**, or **xlExcel4IntlMacroSheet**. The default value is **xlWorksheet**.

Remarks

If ***Before*** and ***After*** are both omitted, the new sheet is inserted before the active sheet.

Add Method (Sheets Collection) Example

This example inserts a new worksheet before the last worksheet in the active workbook.

```
ActiveWorkbook.Sheets.Add Before:=Worksheets(Worksheets.Count)
```

Add Method (Styles Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddStylesObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddStylesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddStylesObjA "}

Creates a new style and adds it to the list of styles that are available for the current workbook. Returns a **Style** object.

Syntax

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

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

Name Required **String**. The new style name.

BasedOn Optional **Variant**. A **Range** object that refers to a cell that's used as the basis for the new style. If this argument is omitted, the newly created style is based on the Normal style.

Remarks

If a style with the specified name already exists, this method redefines the existing style based on the cell specified in ***BasedOn***. The following example redefines the Normal style based on the active cell.

```
ActiveWorkbook.Styles.Add Name := "Normal", _
    BasedOn := ActiveCell
```

Add Method (Styles Collection) Example

This example defines a new style based on cell A1 on Sheet1.

```
Worksheets("Sheet1").Activate
ActiveWorkbook.Styles.Add Name:="myNewStyle", _
   BasedOn:=ActiveSheet.Range("A1")
```

This example defines a new style that includes only font properties.

```
With ActiveWorkbook.Styles.Add(Name:="theNewStyle")
   .IncludeNumber = False
   .IncludeFont = True
   .IncludeAlignment = False
   .IncludeBorder = False
   .IncludePatterns = False
   .IncludeProtection = False
   .Font.Name = "Arial"
   .Font.Size = 18
End With
```

Add Method (Trendlines Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddTrendlinesObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddTrendlinesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddTrendlinesObjA "}

Creates a new trendline. Returns a **Trendline** object.

Syntax

expression.**Add(*Type, Order, Period, Forward, Backward, Intercept, DisplayEquation, DisplayRSquared, Name*)**

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

Type Optional **Variant**. The trendline type. Can be one of the following **XlTrendlineType** constants: **xlLinear**, **xlLogarithmic**, **xlExponential**, **xlPolynomial**, **xlMovingAvg**, or **xlPower**. The default value is **xlLinear**.

Order Optional **Variant**. Required if ***Type*** is **xlPolynomial**. The trendline order. Must be an integer from 2 to 6, inclusive.

Period Optional **Variant**. Required if ***Type*** is **xlMovingAvg**. The trendline period. Must be an integer greater than 1 and less than the number of data points in the series you're adding a trendline to.

Forward Optional **Variant**. The number of periods (or units on a scatter chart) that the trendline extends forward.

Backward Optional **Variant**. The number of periods (or units on a scatter chart) that the trendline extends backward.

Intercept Optional **Variant**. The trendline intercept. If this argument is omitted, the intercept is automatically set by the regression.

DisplayEquation Optional **Variant**. **True** to display the equation of the trendline on the chart (in the same data label as the R-squared value). The default value is **False**.

DisplayRSquared Optional **Variant**. **True** to display the R-squared value of the trendline on the chart (in the same data label as the equation). The default value is **False**.

Name Optional **Variant**. The name of the trendline as text. If this argument is omitted, Microsoft Excel generates a name.

Add Method (Trendlines Collection) Example

This example creates a new linear trendline in Chart1.

```
ActiveWorkbook.Charts("Chart1").SeriesCollection(1).Trendlines.Add
```

Add Method (Workbooks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddWorkbooksObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddWorkbooksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddWorkbooksObjA "}

Creates a new workbook. The new workbook becomes the active workbook. Returns a **Workbook** object.

Syntax

expression.**Add(*Template*)**

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

Template Optional **Variant**. Determines how the new workbook is created. If this argument is a string specifying the name of an existing Microsoft Excel file, the new workbook is created with the specified file as a template. If this argument is a constant, the new workbook contains a single sheet of the specified type. Can be one of the following **XlWBATemplate** constants: **xlWBATChart**, **xlWBATExcel4IntlMacroSheet**, **xlWBATExcel4MacroSheet**, or **xlWBATWorksheet**. If this argument is omitted, Microsoft Excel creates a new workbook with a number of blank sheets (the number of sheets is set by the **SheetsInNewWorkbook** property).

Remarks

If the ***Template*** argument specifies a file, the file name can include a path.

Add Method (Workbooks Collection) Example

This example creates a new workbook.

```
Workbooks.Add
```

Add Method (Worksheets Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddWorksheetsObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddWorksheetsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddWorksheetsObjA "}

Creates a new worksheet. The new worksheet becomes the active sheet. Returns a **Worksheet** object.

Syntax

expression.**Add(*Before, After, Count, Type*)**

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

Before Optional **Variant**. An object that specifies the sheet before which the new sheet is added.

After Optional **Variant**. An object that specifies the sheet after which the new sheet is added.

Count Optional **Variant**. The number of sheets to be added. The default value is one.

Type Optional **Variant**. The sheet type. Can be one of the following **XlSheetType** constants: **xlWorksheet**, **xlExcel4MacroSheet**, or **xlExcel4IntlMacroSheet**. The default value is **xlWorksheet**.

Remarks

If ***Before*** and ***After*** are both omitted, the new sheet is inserted before the active sheet.

Add Method (Worksheets Collection) Example

This example creates a new worksheet and inserts it before the active sheet.

```
ActiveWorkbook.Worksheets.Add
```

This example adds a new worksheet after the last worksheet in the active workbook.

```
Worksheets.Add.Move after:=Worksheets(Worksheets.Count)
```

Add Method (SeriesCollection Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddSeriesCollectionObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddSeriesCollectionObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddSeriesCollectionObjA "}

Adds one or more new series to the **SeriesCollection** collection.

Syntax

expression.**Add(*Source, Rowcol, SeriesLabels, CategoryLabels, Replace*)**

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

Source Required **Variant**. The new data, either as a **Range** object or an array of data points.

Rowcol Optional **Variant**. Specifies whether the new values are in the rows or columns of the specified range. Can be one of the following **XlRowCol** constants: **xlRows** or **xlColumns**. The default value is **xlColumns**.

SeriesLabels Optional **Variant**. Ignored if ***Source*** is an array. **True** if the first row or column contains the name of the data series. **False** if the first row or column contains the first data point of the series. If this argument is omitted, Microsoft Excel attempts to determine the location of the series name from the contents of the first row or column.

CategoryLabels Optional **Variant**. Ignored if ***Source*** is an array. **True** if the first row or column contains the name of the category labels. **False** if the first row or column contains the first data point of the series. If this argument is omitted, Microsoft Excel attempts to determine the location of the category label from the contents of the first row or column.

Replace Optional **Variant**. If ***CategoryLabels*** is **True** and ***Replace*** is **True**, the specified categories replace the categories that currently exist for the series. If ***Replace*** is **False**, the existing categories will not be replaced. The default value is **False**.

Add Method (SeriesCollection Collection) Example

This example creates a new series in Chart1. The data source for the new series is range B1:B10 on Sheet1.

```
Charts("Chart1").SeriesCollection.Add _
    Source:=ActiveWorkbook.Worksheets("Sheet1").Range("B1:B10")
```

This example creates a new series on the embedded chart on Sheet1.

```
Worksheets("Sheet1").ChartObjects(1).Activate
ActiveChart.SeriesCollection.Add _
    Source:=Worksheets("Sheet1").Range("B1:B10")
```

BuiltIn Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBuiltInC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBuiltInX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBuiltInA "}

True if the style is a built-in style. Read-only **Boolean**.

BuiltIn Property Example

This example creates a list on worksheet one that contains the names and built-in status of all the styles in the active workbook.

```
r = 0
Worksheets(1).Activate
For Each s In ActiveWorkbook.Styles
   r = r + 1
   Cells(r, 1).Value = s.Name
   Cells(r, 2).Value = s.BuiltIn
Next
```

Enabled Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnabledC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnabledX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnabledA "}

True if the object is enabled. Read/write **Boolean**.

Enabled Property Example

This example disables embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Enabled = False
```

Item

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemC "}

Returns a single member of a collection. Read-only.

Item is the default member for a collection. For example, the following two lines of code are equivalent.

```
ActiveWorkbook.Worksheets.Item(1)
ActiveWorkbook.Worksheets(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 or property for a given collection, select the name of the collection in the following list.

AddIns	**Areas**	**Axes**
Borders	**CalculatedFields**	**CalculatedItems**
ChartGroups	**ChartObjects**	**Charts**
Comments	**CustomViews**	**DataLabels**
Dialogs	**FormatConditions**	**GroupShapes**
HPageBreaks	**LegendEntries**	**Names**
ODBCErrors	**OLEObjects**	**Panes**
Parameters	**PivotCaches**	**PivotFields**
PivotFormulas	**PivotItems**	**PivotTables**
Points	**QueryTables**	**Range**
RecentFiles	**Scenarios**	**SeriesCollection**
ShapeRange	**Shapes**	**Sheets**
Styles	**Trendlines**	**VPageBreaks**
Windows	**Workbooks**	**Worksheets**

Item Property (Borders Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemBordersObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemBordersObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemBordersObjA "}

Returns a **Border** object that represents one of the borders of either a range of cells or a style. Read-only.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns a **Borders** collection.

Index Required **Long**. The border to return. Can be one of the following **XlBorderType** constants: **xlInsideHorizontal**, **xlInsideVertical**, **xlDiagonalDown**, **xlDiagonalUp**, **xlEdgeBottom**, **xlEdgeLeft**, **xlEdgeRight**, or **xlEdgeTop**.

Item Property (Borders Collection) Example

This following example sets the color of the bottom border of cells A1:G1.

```
Worksheets("Sheet1").Range("a1:g1"). _
    Borders.Item(xlEdgeBottom).Color = RGB(255, 0, 0)
```

Item Property (Range Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemRangeObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemRangeObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemRangeObjA "}

Returns a **Range** object that represents a range at an offset to the specified range. Read-only.

Syntax 1

expression.**Item(*RowIndex, ColumnIndex*)**

Syntax 2

expression.**Item(*RowIndex*)**

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

RowIndex Syntax 1: Required **Variant**. The row number of the cell you want to access, starting with 1 for the first row in the range.

Syntax 2: Required **Variant**. The index number of the cell you want to access, in order from left to right, then down. `Range.Item(1)` returns the upper-left cell in the range; `Range.Item(2)` returns the cell immediately to the right of the upper-left cell.

ColumnIndex Optional **Variant**. A number or string that indicates the column number of the cell you want to access, starting with either 1 or "A" for the first column in the range.

Remarks

Syntax 1 uses a row number and a column number or letter as index arguments. For more information about this syntax, see the **Range** object.The ***RowIndex*** and ***ColumnIndex*** arguments are relative offsets. In other words, specifying a ***RowIndex*** of 1 returns cells in the first row of the range, not the first row of the worksheet. For example, if the selection is cell C3, `Selection.Cells(2, 2)` returns cell D4 (you can use the **Item** property to index outside the original range).

Item Property (Range Object) Example

This example fills the range A1:A10 on Sheet1, based on the contents of cell A1.

```
Worksheets("Sheet1").Range.Item("A1:A10").FillDown
```

Item Property (Dialogs Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemDialogsObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemDialogsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemDialogsObjA "}

Returns a **Dialog** object that represents a single built-in dialog box. Read-only.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns a **Dialogs** collection.

Index Required **Long**. The built-in dialog box to return. Can be any one of the **XlBuiltInDialog** constants.

Remarks

Using the **Item** property of the **Dialogs** collection and the **Show** method, you can display approximately 200 built-in dialog boxes. Each dialog box has a constant assigned to it; these constants all begin with "xlDialog."

For a table of the available constants and their corresponding argument lists, see Built-In Dialog Box Argument Lists.

The **Item** property of the **Dialogs** collection may fail if you try to show a dialog box in an incorrect context. For example, to display the **Data Labels** dialog box (using the Visual Basic expression `Application.Dialogs(xlDialogDataLabel).Show`), the active sheet must be a chart; otherwise, the property fails.

Item Property (Dialogs Collection) Example

This example displays the **Open** dialog box and selects the **Read-Only** option.

```
Application.Dialogs.Item(xlDialogOpen).Show arg3:=True
```

SQLBind Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLBindC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLBindX":1}

Don't use **SQLBind** and the other ODBC functions in the Xlodbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLBind specifies where results are placed when they're retrieved with **SQLRetrieve** or **SQLRetrieveToFile**. Use **SQLBind** to change the column order of the result set from a query, or to place the result set columns in nonadjacent worksheet columns.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLBind(*ConnectionNum, Column, Reference*)

ConnectionNum Required. The unique connection ID (returned by **SQLOpen**) of the data source for which you want to bind results.

Column Optional. The column number of the result set you want to bind. Columns in the result set are numbered from left to right, starting with 1. If you omit ***Column***, all bindings for ***ConnectionNum*** are removed.

Column 0 (zero) contains row numbers for the result set. You can return the row numbers by binding column 0 (zero).

Reference Optional. A **Range** object that specifies the location of a single cell on a worksheet where you want the results to be bound. If ***Reference*** is omitted, binding is removed for the column.

Return Value

This function returns an array that lists the bound columns for the current connection, by column number.

If **SQLBind** is unable to bind the column to the cell in the specified reference, it returns Error 2042.

If ***ConnectionNum*** isn't valid or if you try to bind a cell that isn't available, **SQLBind** returns Error 2015.

If ***Reference*** refers to more than a single cell, **SQLBind** returns Error 2023.

If **SQLRetrieve** doesn't have a destination parameter, **SQLBind** places the result set in the location indicated by ***Reference***.

Remarks

SQLBind tells the ODBC Control Panel Administrator where to place results when they're received by way of **SQLRetrieve** The results are placed in the reference cell and the cells immediately below it.

Use **SQLBind** if you want the results from different columns to be placed in disjoint worksheet areas.

Use **SQLBind** for each column in the result set. A binding remains valid as long as the connection specified by ***ConnectionNum*** is open.

Call **SQLBind** after you call **SQLOpen** and **SQLExecQuery**, but before you call **SQLRetrieve** or **SQLRetrieveToFile**. Calls to **SQLBind** don't affect results that have already been retrieved.

SQLBind Function Example

This example runs a query on the NWind sample database, and then it uses the **SQLBind** function to display only the fourth and ninth columns of the query result set (the product name and the quantity on order) on Sheet1.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
queryString = "SELECT * FROM product.dbf WHERE (product.ON_ORDER<>0)"
chan = SQLOpen("DSN=" & databaseName)
SQLExecQuery chan, queryString
Set output1 = Worksheets("Sheet1").Range("A1")
Set output2 = Worksheets("Sheet1").Range("B1")
SQLBind chan, 4, output1
SQLBind chan, 9, output2
SQLRetrieve chan
SQLClose chan
```

SQLClose Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLCloseC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLCloseX":1}

Don't use **SQLClose** and the other ODBC functions in the Xlodbbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLClose closes a connection to an external data source.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLClose(*ConnectionNum*)

ConnectionNum Required. The unique connection ID of the data source you want to disconnect from.

Return Value

If the connection is successfully closed, this function returns 0 (zero) and the connection ID is no longer valid.

If ***ConnectionNum*** isn't valid, this function returns Error 2015.

If **SQLClose** is unable to disconnect from the data source, it returns Error 2042.

SQLClose Function Example

This example runs a query on the NWind sample database. The result of the query, displayed on Sheet1, is a list of all products that are currently on order.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
queryString = "SELECT * FROM product.dbf WHERE (product.ON_ORDER<>0)"
chan = SQLOpen("DSN=" & databaseName)
SQLExecQuery chan, queryString
Set output = Worksheets("Sheet1").Range("A1")
SQLRetrieve chan, output, , , True
SQLClose chan
```

SQLError Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLErrorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLErrorX":1}

Don't use **SQLError** and the other ODBC functions in the Xlodbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLError returns detailed error information when it's called after one of the other ODBC functions fails. If **SQLError** itself fails, it cannot return error information.

Error information is defined and stored in memory whenever an ODBC function fails. To make the error information available, call the **SQLError** function.

SQLError provides detailed error information only about errors that occur when an ODBC function fails. It doesn't provide information about Microsoft Excel errors.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLError()

Return Value

If there are errors, **SQLError** returns detailed error information in a two-dimensional array in which each row describes one error.

Each row has the following three fields for information obtained through the **SQLError** function call in ODBC:

- A character string that indicates the ODBC error class and subclass
- A numeric value that indicates the data source native error code.
- A text message that describes the error.

If a function call generates multiple errors, **SQLError** creates a row for each error.

If there are no errors from a previous ODBC function call, this function returns only Error 2042.

SQLError Function Example

This example generates an intentional error by attempting to open a connection to the NWind sample database by using an incorrect connection string (NWind is misspelled). The error information is displayed on Sheet1.

```
chan = SQLOpen("DSN=NWin")
returnArray = SQLError()
For i = LBound(returnArray, 1) To UBound(returnArray, 1)
   Worksheets("Sheet1").Cells(1, i).Formula = returnArray(i)
Next i
SQLClose chan
```

SQLExecQuery Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLExecQueryC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLExecQueryX":1}

Don't use **SQLExecQuery** and the other ODBC functions in the XLODBC.XLA add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLExecQuery executes a query on a data source with a connection that has been established with **SQLOpen**.

SQLExecQuery executes only the query. Use **SQLRetrieve** or **SQLRetrieveToFile** to get the results.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLExecQuery(*ConnectionNum, QueryText*)

ConnectionNum Required. The unique connection ID returned by **SQLOpen** that identifies the data source you want to query.

QueryText Required. The query to be executed on the data source. The query must follow the SQL syntax guidelines for the specific driver.

Return Value

The value returned by **SQLExecQuery** depends on the SQL statement, as shown in the following table.

SQL statement	Return value
SELECT	The number of columns in the result set
UPDATE, INSERT, or DELETE	The number of rows affected by the statement
Any other valid SQL statement	0 (zero)

If **SQLExecQuery** is unable to execute the query on the specified data source, it returns Error 2042.

If ***ConnectionNum*** isn't valid, **SQLExecQuery** returns Error 2015.

Remarks

Before calling **SQLExecQuery**, you must establish a connection to a data source by using **SQLOpen** The unique connection ID returned by **SQLOpen** is used by **SQLExecQuery** to send queries to the data source.

If you call **SQLExecQuery** using a previously used connection ID, any pending results on that connection are replaced by the new results.

SQLExecQuery Function Example

This example runs a query on the NWind sample database. The result of the query, displayed on Sheet1, is a list of all products that are currently on order.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
queryString = "SELECT * FROM product.dbf WHERE (product.ON_ORDER<>0)"
chan = SQLOpen("DSN=" & databaseName)
SQLExecQuery chan, queryString
Set output = Worksheets("Sheet1").Range("A1")
SQLRetrieve chan, output, , , True
SQLClose chan
```

SQLGetSchema Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLGetSchemaC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLGetSchemaX":1}

Don't use **SQLGetSchema** and the other ODBC functions in the Xlodbbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLGetSchema returns information about the structure of the data source on a particular connection.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLGetSchema(*ConnectionNum, TypeNum, QualifierText*)

ConnectionNum Required. The unique connection ID of the data source you connected to by using **SQLOpen** and for which you want information.

TypeNum Required. Specifies the type of information you want returned, as shown in the following table.

Value	Meaning
1	A list of all the available data sources.
2	A list of databases on the current connection.
3	A list of owners in a database on the current connection.
4	A list of tables for a given owner and database on the current connection.
5	A list of columns in a particular table and their ODBC SQL data types, in a two-dimensional array. The first field contains the name of the column; the second field is the column's ODBC SQL data type.
6	The user ID of the current user.
7	The name of the current database.
8	The name of the data source defined during setup or defined by using the ODBC Control Panel Administrator.
9	The name of the DBMS that the data source uses – for example, ORACLE or SQL Server.
10	The server name for the data source.
11	The terminology used by the data source to refer to the owners – for example "owner", "Authorization ID", or "Schema".
12	The terminology used by the data source to refer a table – for example, "table" or "file".
13	The terminology used by the data source to refer to a qualifier – for example, "database" or "folder".
14	The terminology used by the data source to refer to a procedure – for example, "database procedure", "stored procedure", or "procedure".

QualifierText Optional. Included only for the ***TypeNum*** values 3, 4, and 5. A string that qualifies the search, as shown in the following table.

TypeNum	*QualifierText*
3	The name of the database in the current data source. **SQLGetSchema** returns the names of the table owners in that database.
4	Both a database name and an owner name. The syntax consists of the database name followed by the owner's name, with a period separating the two; for example, "DatabaseName.OwnerName". This function returns an array of table names that are located in the given database and owned by the given owner.
5	The name of a table. **SQLGetSchema** returns information about the columns in the table.

Return Value

The return value from a successful call to **SQLGetSchema** depends on the type of information that's requested.

If **SQLGetSchema** cannot find the requested information, it returns Error 2042.

If ***ConnectionNum*** isn't valid, this function returns Error 2015.

Remarks

SQLGetSchema uses the ODBC API functions **SQLGetInfo** and **SQLTables** to find the requested information.

SQLGetSchema Function Example

This example retrieves the database name and DBMS name for the NWind sample database and then displays these names in a message box.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
chan = SQLOpen("DSN=" & databaseName)
dsName = SQLGetSchema(chan, 8)
dsDBMS = SQLGetSchema(chan, 9)
MsgBox "Database name is " & dsName & ", and its DBMS is " & dsDBMS
SQLClose chan
```

SQLOpen Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLOpenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLOpenX":1}

Don't use **SQLOpen** and the other ODBC functions in the Xlodbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLOpen establishes a connection to a data source.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLOpen(*ConnectionStr, OutputRef, DriverPrompt*)

ConnectionStr Required. Supplies the information required by the driver being used to connect to a data source; must follow the driver's format.

ConnectionStr supplies the data source name and other information − such as user ID and passwords − that the driver requires to make a connection.

You must define the data source name (DSN) used in ***ConnectionStr*** before you try to connect to it.

OutputRef Optional. A **Range** object (must be a single cell) that contains the completed connection string.

Use ***OutputRef*** when you want **SQLOpen** to return the completed connection string to a worksheet.

DriverPrompt Optional. Specifies whether the driver dialog box is displayed and, if it is, which options are available in it. Use one of the values described in the following table. If ***DriverPrompt*** is omitted, **SQLOpen** uses 2 as the default.

Value	Meaning
1	The driver dialog box is always displayed.
2	The driver dialog box is displayed only if information provided by the connection string and the data source specification aren't sufficient to complete the connection. All dialog box options are available.
3	The same as 2 except that dialog box options that aren't required are dimmed (unavailable).
4	The driver dialog box isn't displayed. If the connection isn't successful, SQLOpen returns an error.

Return Value

If successful, **SQLOpen** returns a unique connection ID number. Use the connection ID number with the other ODBC functions.

If **SQLOpen** is unable to connect using the information you provide, it returns Error 2042. Additional error information is placed in memory for use by **SQLError**.

SQLOpen Function Example

This example runs a query on the NWind sample database. The result of the query, displayed on Sheet1, is a list of all products that are currently on order.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
queryString = "SELECT * FROM product.dbf WHERE (product.ON_ORDER<>0)"
chan = SQLOpen("DSN=" & databaseName)
SQLExecQuery chan, queryString
Set output = Worksheets("Sheet1").Range("A1")
SQLRetrieve chan, output, , , True
SQLClose chan
```

SQLRequest Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLRequestC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLRequestX":1}

Don't use **SQLRequest** and the other ODBC functions in the Xlodbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLRequest connects to an external data source and runs a query from a worksheet, and then it returns the result of the query as an array.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLRequest(*ConnectionStr, QueryText, OutputRef, DriverPrompt, ColNamesLogical*)

ConnectionStr Required. Supplies information – such as the data source name, user ID, and passwords – required by the driver being used to connect to a data source; must follow the driver's format.

You must define the data source name (DSN) used in ***ConnectionStr*** before you try to connect to it.

If **SQLRequest** is unable to access the data source using ***ConnectionStr***, it returns Error 2042.

QueryText Required. The SQL statement you want to execute on the data source.

If **SQLRequest** is unable to execute ***QueryText*** on the specified data source, it returns Error 2042.

OutputRef Optional. A **Range** object (must be a single cell) where you want the completed connection string to be placed.

DriverPrompt Optional. Specifies whether the driver dialog box is displayed and which options are available. Use one of the values described in the following table. If ***DriverPrompt*** is omitted, **SQLRequest** uses 2 as the default.

Value	Meaning
1	The driver dialog box is always displayed.
2	The driver dialog box is displayed only if information provided by the connection string and the data source specification isn't sufficient to complete the connection. All dialog box options are available.
3	The driver dialog box is displayed only if information provided by the connection string and the data source specification isn't sufficient to complete the connection. Dialog box options that aren't required are dimmed (unavailable).
4	The dialog box isn't displayed. If the connection isn't successful, it returns an error.

ColNamesLogical Optional. **True** to have the column names be returned as the first row of results. **False** to not have the column names be returned. If ***ColNamesLogical*** is omitted, the default value is **False**.

Remarks

The arguments to the **SQLRequest** function are in a different order than the arguments to the SQL.REQUEST macro function.

Return Value

If this function completes all of its actions, it returns an array of query results or the number of rows affected by the query.

If **SQLRequest** is unable to complete all of its actions, it returns an error value and places the error information in memory for **SQLError**.

If **SQLRequest** is unable to access the data source using ***connectionStr***, it returns Error 2042.

SQLRequest Function Example

This example runs a query on the NWind sample database. The result of the query, displayed on Sheet1, is a list of all products that are currently on order. The **SQLRequest** function also writes the full connection string to Sheet2.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
queryString = "SELECT * FROM product.dbf WHERE (product.ON_ORDER<>0)"
returnArray = SQLRequest("DSN=" & databaseName, _
         queryString, _
         Worksheets("Sheet1").Range("A1"), _
         2, True)
For i = LBound(returnArray, 1) To UBound(returnArray, 1)
   For j = LBound(returnArray, 2) To UBound(returnArray, 2)
      Worksheets("Sheet1").Cells(i, j).Formula = returnArray(i, j)
   Next j
Next i
```

SQLRetrieve Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLRetrieveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLRetrieveX":1}

Don't use **SQLRetrieve** and the other ODBC functions in the Xlodbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLRetrieve retrieves all or part of the results from a previously executed query.

Before using **SQLRetrieve**, you must establish a connection with **SQLOpen**, execute a query with **SQLExecQuery**, and have the results pending.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLRetrieve(*ConnectionNum, DestinationRef, MaxColumns, MaxRows, ColNamesLogical, RowNumsLogical, NamedRngLogical, FetchFirstLogical*)

ConnectionNum Required. The unique connection ID returned by **SQLOpen** and for which you have pending query results that were generated by **SQLExecQuery**.

If ***ConnectionNum*** isn't valid, **SQLExecQuery** returns Error 2015.

DestinationRef Optional. A **Range** object that specifies where the results should be placed. This function overwrites any values in the cells, without confirmation.

If ***DestinationRef*** refers to a single cell, **SQLRetrieve** returns all the pending results in that cell and in the cells to the right of and below it.

If ***DestinationRef*** is omitted, the bindings established by previous calls to **SQLBind** are used to return results. If no bindings exist for the current connection, **SQLRetrieve** returns Error 2023.

If a particular result column hasn't been bound and ***DestinationRef*** is omitted, the results are discarded.

MaxColumns Optional. The maximum number of columns returned to the worksheet, starting at ***DestinationRef***.

If ***MaxColumns*** specifies more columns than are available in the result, **SQLRetrieve** places data in the columns for which data is available and clears the additional columns.

If ***MaxColumns*** specifies fewer columns than are available in the result, **SQLRetrieve** discards the rightmost result columns until the results fit the specified size.

The order in which the data source returns the columns determines column position.

If ***MaxColumns*** is omitted, all the results are returned.

MaxRows Optional. The maximum number of rows to be returned to the worksheet, starting at ***DestinationRef***.

If ***MaxRows*** specifies more rows than are available in the results, **SQLRetrieve** places data in the rows for which data is available and clears the additional rows.

If ***MaxRows*** specifies fewer rows than are available in the results, **SQLRetrieve** places data in the selected rows but doesn't discard the additional rows. You can retrieve extra rows by using **SQLRetrieve** again and setting ***FetchFirstLogical*** to **False**.

If ***MaxRows*** is omitted, all the rows in the results are returned.

ColNamesLogical Optional. **True** to have the column names be returned as the first row of results. **False** or omitted to have the column names not be returned.

RowNumsLogical Optional. Used only when ***DestinationRef*** is included in the function call. **True** to have the first column in the result set contain row numbers. **False** or omitted to have the row numbers not be returned. You can also retrieve row numbers by binding column 0 (zero) with

SQLBind.

NamedRngLogical Optional. **True** to have each column of the results be declared as a named range on the worksheet. The name of each range is the result column name. The named range includes only the rows that are returned with **SQLRetrieve**. The default value is **False**.

FetchFirstLogical Optional. Allows you to request results from the beginning of the result set. If ***FetchFirstLogical*** is **False**, **SQLRetrieve** can be called repeatedly to return the next set of rows until all the result rows have been returned. When there are no more rows in the result set, **SQLRequest** returns 0 (zero). If you want to retrieve results from the beginning of the result set, set ***FetchFirstLogical*** to **True**. To retrieve additional rows from the result set, set ***FetchFirstLogical*** to **False** in subsequent calls. The default value is **False**.

Return Value

SQLRetrieve returns the number of rows in the result set.

If **SQLRetrieve** is unable to retrieve the results on the specified data source or if there are no results pending, it returns Error 2042. If no data is found, **SQLRetrieve** returns 0 (zero).

Remarks

Before calling **SQLRetrieve**, you must do the following:

1. Establish a connection with a data source by using **SQLOpen**.
2. Use the connection ID returned in **SQLOpen** to send a query with **SQLExecQuery**.

SQLRetrieve Function Example

This example runs a query on the NWind sample database. The result of the query, displayed on Sheet1, is a list of all products that are currently on order.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
queryString = "SELECT * FROM product.dbf WHERE (product.ON_ORDER<>0)"
chan = SQLOpen("DSN=" & databaseName)
SQLExecQuery chan, queryString
Set output = Worksheets("Sheet1").Range("A1")
SQLRetrieve chan, output, , , True
SQLClose chan
```

SQLRetrieveToFile Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSQLRetrieveToFileC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSQLRetrieveToFileX":1}

Don't use **SQLRetrieveToFile** and the other ODBC functions in the Xlodbc.xla add-in; use the objects, methods, and properties in the Data Access Objects (DAO) library instead. For more information about Data Access Objects, see Data Access Overview.

SQLRetrieveToFile retrieves all the results from a previously executed query and places them in a file.

To use this function, you must have established a connection with a data source by using **SQLOpen**, executed a query by using **SQLExecQuery**, and have the results of the query pending.

This function is contained in the Xlodbc.xla add-in (ODBC Add-In on the Macintosh). Before you use the function, you must establish a reference to the add-in by using the **References** command (**Tools** menu).

Syntax

SQLRetrieveToFile(*ConnectionNum, Destination, ColNamesLogical, ColumnDelimiter*)

ConnectionNum Required. The unique connection ID returned by **SQLOpen** and for which you have pending query results that were generated by **SQLExecQuery**.

If ***ConnectionNum*** isn't valid, **SQLExecQuery** returns Error 2015.

Destination Required. A string that specifies the name and path of the file where you want to place the results. If the file exists, its contents are replaced with the query results. If the file doesn't exist, **SQLRetrieveToFile** creates and opens the file and fills it with the results.

The format of the data in the file is compatible with the Microsoft Excel .csv (comma-separated value) file format.

Columns are separated by the character specified by ***ColumnDelimiter***, and the individual rows are separated by a carriage return.

If the file specified by ***Destination*** cannot be opened, **SQLRetrieveToFile** returns Error 2042.

ColNamesLogical Optional. **True** to have the column names be returned as the first row of data. **False** or omitted to have the column names not be returned.

ColumnDelimiter Optional. A string that specifies the character used to separate the elements in each row. For example, use "," to specify a comma delimiter, or use ";" to specify a semicolon delimiter. If you omit ***ColumnDelimiter***, the list separator character is used.

Return Value

If successful, **SQLRetrieveToFile** returns the query results, writes them to a file, and then returns the number of rows that were written to the file.

If **SQLRetrieveToFile** is unable to retrieve the results, it returns Error 2042 and doesn't write the file.

If there are no pending results on the connection, **SQLRetrieveToFile** returns Error 2042.

Remarks

Before calling **SQLRetrieveToFile**, you must do the following:

1. Establish a connection with a data source by using **SQLOpen**.
2. Use the connection ID returned by **SQLOpen** to send a query with **SQLExecQuery**.

SQLRetrieveToFile Function Example

This example runs a query on the NWind sample database. The result of the query, which is a list of all products that are currently on order, is written as the delimited text file Output.txt in the current folder.

```
If Application.OperatingSystem Like "*Win*" Then
   databaseName = "NWind"
Else     'Macintosh
   databaseName = "NorthWind"
End If
queryString = "SELECT * FROM product.dbf WHERE (product.ON_ORDER<>0)"
chan = SQLOpen("DSN=" & databaseName)
SQLExecQuery chan, queryString
SQLRetrieveToFile chan, "OUTPUT.TXT", True
SQLClose chan
```

SolverAdd Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverAddC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverAddX":1}

Adds a constraint to the current problem. Equivalent to clicking **Solver** on the **Tools** menu and then clicking **Add** in the **Solver Parameters** dialog box.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverAdd(*CellRef, Relation, FormulaText*)

CellRef Required **Variant**. A reference to a cell or a range of cells that forms the left side of a constraint.

Relation Required **Integer**. The arithmetic relationship between the left and right sides of the constraint. If you choose 4 or 5, ***CellRef*** must refer to adjustable (changing) cells, and ***FormulaText*** shouldn't be specified.

Relation	Arithmetic relationship
1	<=
2	=
3	>=
4	Cells referenced by ***CellRef*** must have final values that are integers.
5	Cells referenced by ***CellRef*** must have final values of either 0 (zero) or 1.

FormulaText Optional **Variant**. The right side of the constraint.

Remarks

After constraints are added, you can manipulate them with the **SolverChange** and **SolverDelete** functions.

SolverAdd Function Example

This example uses the Solver functions to maximize gross profit in a business problem. The **SolverAdd** function is used to add three constraints to the current problem.

```
Worksheets("Sheet1").Activate
SolverReset
SolverOptions precision:=0.001
SolverOK setCell:=Range("TotalProfit"), _
   maxMinVal:=1, _
   byChange:=Range("C4:E6")
SolverAdd cellRef:=Range("F4:F6"), _
   relation:=1, _
   formulaText:=100
SolverAdd cellRef:=Range("C4:E6"), _
   relation:=3, _
   formulaText:=0
SolverAdd cellRef:=Range("C4:E6"), _
   relation:=4
SolverSolve userFinish:=False
SolverSave saveArea:=Range("A33")
```

SolverChange Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverChangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverChangeX":1}

Changes an existing constraint. Equivalent to clicking **Solver** on the **Tools** menu and then clicking **Change** in the **Solver Parameters** dialog box.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverChange(*CellRef, Relation, FormulaText*)

CellRef Required **Variant**. A reference to a cell or a range of cells that forms the left side of a constraint.

Relation Required **Integer**. The arithmetic relationship between the left and right sides of the constraint. If you choose 4 or 5 , ***CellRef*** must refer to adjustable (changing) cells, and ***FormulaText*** shouldn't be specified.

Relation	Arithmetic relationship
1	<=
2	=
3	>=
4	Cells referenced by ***CellRef*** must have final values that are integers.
5	Cells referenced by ***CellRef*** must have final values of either 0 (zero) or 1.

FormulaText Optional **Variant**. The right side of the constraint.

Remarks

If ***CellRef*** and ***Relation*** don't match an existing constraint, you must use the **SolverDelete** and **SolverAdd** functions to change the constraint.

SolverChange Function Example

This example loads the previously calculated Solver model stored on Sheet1, changes one of the constraints, and then solves the model again.

```
Worksheets("Sheet1").Activate
SolverLoad loadArea:=Range("A33:A38")
SolverChange cellRef:=Range("F4:F6"), _
   relation:=1, _
   formulaText:=200
SolverSolve userFinish:=False
```

SolverDelete Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverDeleteC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverDeleteX":1}

Deletes an existing constraint. Equivalent to clicking **Solver** on the **Tools** menu and then clicking **Delete** in the **Solver Parameters** dialog box.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverDelete(*CellRef, Relation, FormulaText*)

CellRef Required **Variant**. Reference to a cell or a range of cells that forms the left side of a constraint.

Relation Required **Integer**. The arithmetic relationship between the left and right sides of the constraint. If you choose 4 or 5, ***CellRef*** must refer to adjustable (changing) cells, and ***FormulaText*** shouldn't be specified.

Relation	Arithmetic relationship
1	<=
2	=
3	>=
4	Cells referenced by ***CellRef*** must have final values that are integers.
5	Cells referenced by ***CellRef*** must have final values of either 0 (zero) or 1.

FormulaText Optional **Variant**. The right side of the constraint.

SolverDelete Function Example

This example loads the previously calculated Solver model stored on Sheet1, deletes one of the constraints, and then solves the model again.

```
Worksheets("Sheet1").Activate
SolverLoad loadArea:=Range("A33:A38")
SolverDelete cellRef:=Range("C4:E6"), _
   relation:=4
SolverSolve userFinish:=False
```

SolverFinish Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverFinishC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverFinishX":1}

Tells Microsoft Excel what to do with the results and what kind of report to create when the solution process is completed.

Before you use this function, you must establish a reference to the Solver add-in. With the Visual Basic Editor active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the Library\Solver folder.

Syntax

SolverFinish(*KeepFinal, ReportArray*)

KeepFinal Optional **Variant**. Can be either 1 or 2. If ***KeepFinal*** is 1 or omitted, the final solution values are kept in the changing cells, replacing any former values. If ***KeepFinal*** is 2, the final solution values are discarded, and the former values are restored.

ReportArray Optional **Variant**. The kind of report Microsoft Excel will create when Solver is finished: 1 creates an answer report, 2 creates a sensitivity report, and 3 creates a limit report. Use the **Array** function to specify the reports you want to display – for example, `ReportArray:= Array(1,3)`.

SolverFinish Function Example

This example loads the previously calculated Solver model stored on Sheet1, solves the model again, and then generates an answer report on a new worksheet.

```
Worksheets("Sheet1").Activate
SolverLoad loadArea:=Range("A33:A38")
SolverSolve userFinish:=True
SolverFinish keepFinal:=1, reportArray:=Array(1)
```

SolverFinishDialog Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverFinishDialogC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverFinishDialogX":1}

Tells Microsoft Excel what to do with the results and what kind of report to create when the solution process is completed. Equivalent to the **SolverFinish** function, but also displays the **Solver Results** dialog box after solving a problem.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverFinishDialog(*KeepFinal, ReportArray*)

KeepFinal Optional **Variant**. Can be either 1 or 2. If ***KeepFinal*** is 1 or omitted, the final solution values are kept in the changing cells, replacing any former values. If ***KeepFinal*** is 2, the final solution values are discarded, and the former values are restored.

ReportArray Optional **Variant**. The kind of report Microsoft Excel will create when Solver is finished: 1 creates an answer report, 2 creates a sensitivity report, and 3 creates a limit report. Use the **Array** function to specify the reports you want to display – for example, `ReportArray:= Array(1,3)`.

SolverFinishDialog Function Example

This example loads the previously calculated Solver model stored on Sheet1, solves the model again, and then displays the **Finish** dialog box with two preset options.

```
Worksheets("Sheet1").Activate
SolverLoad loadArea:=Range("A33:A38")
SolverSolve userFinish:=True
SolverFinishDialog keepFinal:=1, reportArray:=Array(1)
```

SolverGet Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverGetC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverGetX":1}

Returns information about current settings for Solver. The settings are specified in the **Solver Parameters** and **Solver Options** dialog boxes.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverGet(*TypeNum, SheetName*)

TypeNum Required **Integer**. A number specifying the type of information you want. The following settings are specified in the **Solver Parameters** dialog box.

TypeNum	**Returns**
1	The reference in the **Set Target Cell** box, or the #N/A error value if Solver hasn't been used on the active sheet.
2	A number corresponding to the **Equal To** option: 1 represents Max, 2 represents Min, and 3 represents Value Of.
3	The value in the **Value Of** box.
4	The reference (as a multiple reference, if necessary) in the **By Changing Cells** box.
5	The number of constraints.
6	An array of the left sides of the constraints, in text form.
7	An array of numbers corresponding to the relationships between the left and right sides of the constraints: 1 represents <=, 2 represents =, 3 represents >=, 4 represents int, and 5 represents bin.
8	An array of the right sides of the constraints, in text form.

The following settings are specified in the **Solver Options** dialog box.

TypeNum	**Returns**
9	The maximum calculation time.
10	The maximum number of iterations.
11	The precision.
12	The integer tolerance value.
13	**True** if the **Assume Linear Model** check box is selected; **False** if it's cleared.
14	**True** if the **Show Iteration Results** check box is selected; **False** if it's cleared.
15	**True** if the **Use Automatic Scaling** check box is selected; **False** if it's cleared.
16	A number corresponding to the type of estimates: 1 represents Tangent, and 2 represents Quadratic.
17	A number corresponding to the type of derivatives: 1 represents Forward, and 2 represents Central.
18	A number corresponding to the type of search: 1

	represents Quasi-Newton, and 2 represents Conjugate Gradient.
19	The convergence value.
20	**True** if the **Assume Non-Negative** check box is selected.

SheetName Optional **Variant**. The name of the sheet that contains the Solver model for which you want information. If ***SheetName*** is omitted, this sheet is assumed to be the active sheet.

SolverGet Function Example

This example displays a message if you haven't used Solver on Sheet1.

```
Worksheets("Sheet1").Activate
state = SolverGet(typeNum:=1)
If IsError(state) Then
   MsgBox "You have not used Solver on the active sheet"
End If
```

SolverLoad Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverLoadC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverLoadX":1}

Loads existing Solver model parameters that have been saved to the worksheet.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverLoad(*LoadArea*)

LoadArea Required **Variant**. A reference on the active worksheet to a range of cells from which you want to load a complete problem specification. The first cell in the ***LoadArea*** contains a formula for the **Set Target Cell** box in the **Solver Parameters** dialog box; the second cell contains a formula for the **By Changing Cells** box; subsequent cells contain constraints in the form of logical formulas. The last cell optionally contains an array of Solver option values. For more information, see **SolverOptions**. The range represented by the argument ***LoadArea*** can be on any worksheet, but you must specify the worksheet if it's not the active sheet. For example, `SolverLoad("Sheet2!A1:A3")` loads a model from Sheet2 even if it's not the active sheet.

SolverLoad Function Example

This example loads the previously calculated Solver model stored on Sheet1, changes one of the constraints, and then solves the model again.

```
Worksheets("Sheet1").Activate
SolverLoad loadArea:=Range("A33:A38")
SolverChange cellRef:=Range("F4:F6"), _
   relation:=1, _
   formulaText:=200
SolverSolve userFinish:=False
```

SolverOk Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverOkC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverOkX":1}

Defines a basic Solver model. Equivalent to clicking **Solver** on the **Tools** menu and then specifying options in the **Solver Parameters** dialog box.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverOk(*SetCell, MaxMinVal, ValueOf, ByChange*)

SetCell Optional **Variant**. Refers to a single cell on the active worksheet. Corresponds to the **Set Target Cell** box in the **Solver Parameters** dialog box.

MaxMinVal Optional **Variant**. Corresponds to the **Max**, **Min**, and **Value Of** options in the **Solver Parameters** dialog box.

MaxMinVal	**Specifies**
1	Maximize.
2	Minimize.
3	Match a specific value.

ValueOf Optional **Variant**. If ***MaxMinVal*** is 3, you must specify the value to which the target cell is matched.

ByChange Optional **Variant**. The cell or range of cells that will be changed so that you'll obtain the desired result in the target cell. Corresponds to the **By Changing Cells** box in the **Solver Parameters** dialog box.

SolverOK Function Example

This example uses the Solver functions to maximize gross profit in a business problem. The **SolverOK** function defines a problem by specifying the ***SetCell***, ***MaxMinVal***, and ***ByChange*** arguments.

```
Worksheets("Sheet1").Activate
SolverReset
SolverOptions precision:=0.001
SolverOK setCell:=Range("TotalProfit"), _
   maxMinVal:=1, _
   byChange:=Range("C4:E6")
SolverAdd cellRef:=Range("F4:F6"), _
   relation:=1, _
   formulaText:=100
SolverAdd cellRef:=Range("C4:E6"), _
   relation:=3, _
   formulaText:=0
SolverAdd cellRef:=Range("C4:E6"), _
   relation:=4
SolverSolve userFinish:=False
SolverSave saveArea:=Range("A33")
```

SolverOkDialog Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverOkDialogC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverOkDialogX":1}

Same as the **SolverOK** function, but also displays the **Solver** dialog box.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverOkDialog(*SetCell, MaxMinVal, ValueOf, ByChange*)

SetCell Optional **Variant**. Refers to a single cell on the active worksheet. Corresponds to the **Set Target Cell** box in the **Solver Parameters** dialog box.

MaxMinVal Optional **Variant**. Corresponds to the **Max**, **Min**, and **Value Of** options in the **Solver Parameters** dialog box.

MaxMinVal	Specifies
1	Maximize.
2	Minimize.
3	Match a specific value.

ValueOf Optional **Variant**. If ***MaxMinVal*** is 3, you must specify the value that the target cell is matched to.

ByChange Optional **Variant**. The cell or range of cells that will be changed so that you'll obtain the desired result in the target cell. Corresponds to the **By Changing Cells** box in the **Solver Parameters** dialog box.

SolverOKDialog Function Example

This example loads the previously calculated Solver model stored on Sheet1, resets all Solver options, and then displays the **Solver Parameters** dialog box. From this point on, you can use Solver manually.

```
Worksheets("Sheet1").Activate
SolverLoad loadArea:=Range("A33:A38")
SolverReset
SolverOKDialog setCell:=Range("TotalProfit")
SolverSolve userFinish:=False
```

SolverOptions Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverOptionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverOptionsX":1}

Allows you to specify advanced options for your Solver model. This function and its arguments correspond to the options in the **Solver Options** dialog box.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverOptions(*MaxTime, Iterations, Precision, AssumeLinear, StepThru, Estimates, Derivatives, Search, IntTolerance, Scaling, Convergence, AssumeNonNeg*)

MaxTime Optional **Variant**. The maximum amount of time (in seconds) Microsoft Excel will spend solving the problem. The value must be a positive integer. The default value 100 is adequate for most small problems, but you can enter a value as high as 32,767.

Iterations Optional **Variant**. The maximum number of iterations Microsoft Excel will use in solving the problem. The value must be a positive integer. The default value 100 is adequate for most small problems, but you can enter a value as high as 32,767.

Precision Optional **Variant**. A number between 0 (zero) and 1 that specifies the degree of precision to be used in solving the problem. The default precision is 0.000001. A smaller number of decimal places (for example, 0.0001) indicates a lower degree of precision. In general, the higher the degree of precision you specify (the smaller the number), the more time Solver will take to reach solutions.

AssumeLinear Optional **Variant**. **True** to have Solver assume that the underlying model is linear. This speeds the solution process, but it should be used only if all the relationships in the model are linear. The default value is **False**.

StepThru Optional **Variant**. **True** to have Solver pause at each trial solution. You can pass Solver a macro to run at each pause by using the ***ShowRef*** argument of the **SolverSolve** function. **False** to not have Solver pause at each trial solution. The default value if **False**.

Estimates Optional **Variant**. Specifies the approach used to obtain initial estimates of the basic variables in each one-dimensional search: 1 represents tangent estimates, and 2 represents quadratic estimates. Tangent estimates use linear extrapolation from a tangent vector. Quadratic estimates use quadratic extrapolation; this may improve the results for highly nonlinear problems. The default value is 1 (tangent estimates).

Derivatives Optional **Variant**. Specifies forward differencing or central differencing for estimates of partial derivatives of the objective and constraint functions: 1 represents forward differencing, and 2 represents central differencing. Central differencing requires more worksheet recalculations, but it may help with problems that generate a message saying that Solver couldn't improve the solution. With constraints whose values change rapidly near their limits, you should use central differencing. The default value is 1 (forward differencing).

Search Optional **Variant**. Use the ***Search*** options to specify the search algorithm that will be used at each iteration to decide which direction to search in: 1 represents the Newton search method, and 2 represents the conjugate search method. Newton, which uses a quasi-Newton method, is the default search method. This method typically requires more memory than the conjugate search method, but it requires fewer iterations. Conjugate gradient searching requires less memory than the Newton search method, but it typically requires more iterations to reach a particular level of accuracy. You can try this method if you have a large problem and memory usage is a concern. Conjugate searching is especially useful if stepping through the iterations reveals slow progress between successive trial points.

IntTolerance Optional **Variant**. A decimal number between 0 (zero) and 1 that specifies the degree

of integer tolerance. This argument applies only if integer constraints have been defined. You can adjust the tolerance figure, which represents the percentage of error allowed in the optimal solution when an integer constraint is used on any element of the problem. A higher degree of tolerance (allowable percentage of error) would tend to speed up the solution process.

Scaling Optional **Variant**. If two or more constraints differ by several orders of magnitude, **True** to have Solver scale the constraints to similar orders of magnitude during computation. This is useful when the inputs (in the **By Changing Cells** box in the **Solver Parameters** dialog box) and outputs (in the **Set Target Cell** and **Subject to the Constraints** boxes in the **Solver Parameters** dialog box) have large differences in magnitude – for example, maximizing percentage of profit based on million-dollar investments. **False** to have Solver calculate without scaling the constraints. The default value is **False**.

Convergence Optional **Variant**. A number between 0 (zero) and 1 that specifies the degree of convergence tolerance for the nonlinear Solver. When the relative change in the target cell value is less than this tolerance for the last five iterations, Solver stops and displays the message "Solver converged to the current solution. All constraints are satisfied."

AssumeNonNeg Optional **Variant**. **True** to have Solver assume a lower limit of 0 (zero) for all adjustable (changing) cells that don't have explicit lower limits in the **Constraint** list box (the cells must contain nonnegative values). **False** to have Solver use only the limits specified in the **Constraint** list box.

SolverOptions Function Example

This example sets the **Precision** option to .001.

```
Worksheets("Sheet1").Activate
SolverReset
SolverOptions Precision:=0.001
SolverOK SetCell:=Range("TotalProfit"), _
   MaxMinVal:=1, _
   ByChange:=Range("C4:E6")
SolverAdd CellRef:=Range("F4:F6"), _
   Relation:=1, _
   FormulaText:=100
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=3, _
   FormulaText:=0
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=4
SolverSolve UserFinish:=False
SolverSave SaveArea:=Range("A33")
```

SolverSave Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverSaveC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverSaveX":1}

Saves the Solver problem specifications on the worksheet.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverSave(*SaveArea*)

SaveArea Required **Variant**. The range of cells where the Solver model is to be saved. The range represented by the ***SaveArea*** argument can be on any worksheet, but you must specify the worksheet if it's not the active sheet. For example, `SolverSave("Sheet2!A1:A3")` saves the model on Sheet2 even if Sheet2 isn't the active sheet.

SolverSave Function Example

This example uses the Solver functions to maximize gross profit in a business problem. The **SolverSave** function saves the current problem to a range on the active worksheet.

```
Worksheets("Sheet1").Activate
SolverReset
SolverOptions Precision:=0.001
SolverOK SetCell:=Range("TotalProfit"), _
   MaxMinVal:=1, _
   ByChange:=Range("C4:E6")
SolverAdd CellRef:=Range("F4:F6"), _
   Relation:=1, _
   FormulaText:=100
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=3, _
   FormulaText:=0
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=4
SolverSolve UserFinish:=False
SolverSave SaveArea:=Range("A33")
```

SolverSolve Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverSolveC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverSolveX":1}

Begins a Solver solution run. Equivalent to clicking **Solve** in the **Solver Parameters** dialog box.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverSolve(*UserFinish, ShowRef*)

UserFinish Optional **Variant**. **True** to return the results without displaying the **Solver Results** dialog box. **False** or omitted to return the results and display the **Solver Results** dialog box.

ShowRef Optional **Variant**. Used only if **True** is passed to the ***StepThru*** argument of the ***SolverOptions*** function. You can pass the name of a macro (as a string) as the ***ShowRef*** argument. This macro is then called whenever Solver returns an intermediate solution.

SolverSolve Function Example

This example uses the Solver functions to maximize gross profit in a business problem. The **SolverSolve** function begins the Solver solution run.

```
Worksheets("Sheet1").Activate
SolverReset
SolverOptions Precision:=0.001
SolverOK SetCell:=Range("TotalProfit"), _
   MaxMinVal:=1, _
   ByChange:=Range("C4:E6")
SolverAdd CellRef:=Range("F4:F6"), _
   Relation:=1, _
   FormulaText:=100
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=3, _
   FormulaText:=0
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=4
SolverSolve UserFinish:=False
SolverSave SaveArea:=Range("A33")
```

SolverReset Function

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlfctSolverReserC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlfctSolverResetX":1}

Resets all cell selections and constraints in the **Solver Parameters** dialog box and restores all the settings in the **Solver Options** dialog box to their defaults. Equivalent to clicking **Reset All** in the **Solver Parameters** dialog box. The **SolverReset** function is called automatically when you call the **SolverLoad** function.

Before you use this function, you must establish a reference to the Solver add-in. With a Visual Basic module active, click **References** on the **Tools** menu, and then select the **Solver.xla** check box under **Available References**. If **Solver.xla** doesn't appear under **Available References**, click **Browse** and open **Solver.xla** in the \Excel\Library\Solver folder.

Syntax

SolverReset()

SolverReset Function Example

This example resets the Solver settings to their defaults before defining a new problem.

```
Worksheets("Sheet1").Activate
SolverReset
SolverOptions Precision:=0.001
SolverOK SetCell:=Range("TotalProfit"), _
   MaxMinVal:=1, _
   ByChange:=Range("C4:E6")
SolverAdd CellRef:=Range("F4:F6"), _
   Relation:=1, _
   FormulaText:=100
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=3, _
   FormulaText:=0
SolverAdd CellRef:=Range("C4:E6"), _
   Relation:=4
SolverSolve UserFinish:=False
SolverSave SaveArea:=Range("A33")
```

Help topic not available

The Help topic cannot be displayed because Visual Basic for Applications Reference Help cannot be found or wasn't installed.

To install Visual Basic for Applications Reference Help

1. Run Microsoft Office Setup, and then click **Add/Remove**.
2. Click **Microsoft Excel**, and then click **Change Option**.
3. Click **Online Help and sample files**, and then click **Change Option**.
4. Make sure that the **Online Help for Visual Basic** check box is selected.
5. Continue running Setup.

Help topic not available

The Help topic cannot be displayed because Microsoft Excel Help cannot be found or wasn't installed.

To install Microsoft Excel Help

1. Run Microsoft Office Setup, and then click **Add/Remove**.
2. Click **Microsoft Excel**, and then click **Change Option**.
3. Click **Online Help and sample files**, and then click **Change Option**.
4. Make sure that the **Online Help for Microsoft Excel** check box is selected.
5. Continue running Setup.

Help topic not available

The Help topic cannot be displayed because Data Access Objects Help cannot be found or wasn't installed.

To install Data Access Objects Help

1. Run Microsoft Office Setup, and then click **Add/Remove**.
2. Click **Converters, Filters, and Data Access**, and then click **Change Option**.
3. Click **Data Access**, and then click **Change Option**.
4. Make sure that the **Online Help for Data Access Objects** check box is selected.
5. Continue running Setup.

AddFields Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddFieldsA "}

Adds row, column, and page fields to a PivotTable.

Syntax

expression.**AddFields(*RowFields, ColumnFields, PageFields, AddToTable*)**

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

RowFields Optional **Variant**. Specifies a pivot field name (or an array of pivot field names) to be added as rows.

ColumnFields Optional **Variant**. Specifies a pivot field name (or an array of pivot field names) to be added as columns.

PageFields Optional **Variant**. Specifies a pivot field name (or an array of pivot field names) to be added as pages.

AddToTable Optional **Variant**. **True** to add the fields to the PivotTable (none of the existing fields are replaced). **False** to replace existing fields with the new fields. The default value is **False**.

Remarks

You must specify one of the field arguments.

AddFields Method Example

This example replaces the existing column fields in PivotTable one on Sheet1 with the Status and Closed_By fields.

```
Worksheets("Sheet1").PivotTables(1).AddFields _
   ColumnFields:=Array("Status", "Closed_By")
```

BaseField Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBaseFieldC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBaseFieldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBaseFieldA "}

Returns or sets the base field for a custom calculation. Valid only for data fields. Read/write **Variant**.

BaseField Property Example

This example sets the data field in the PivotTable on Sheet1 to calculate the difference from the base field, sets the base field to the field named "ORDER_DATE," and sets the base item to the item named "5/16/89."

```
With Worksheets("Sheet1").Range("A3").PivotField
    .Calculation = xlDifferenceFrom
    .BaseField = "ORDER_DATE"
    .BaseItem = "5/16/89"
End With
```

BaseItem Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBaseItemC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBaseItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBaseItemA "}

Returns or sets the item in the base field for a custom calculation. Valid only for data fields. Read/write **Variant**.

BaseItem Property Example

This example sets the data field in the PivotTable on Sheet1 to calculate the difference from the base field, sets the base field to the field named "ORDER_DATE," and sets the base item to the item named "5/16/89."

```
With Worksheets("Sheet1").Range("A3").PivotField
    .Calculation = xlDifferenceFrom
    .BaseField = "ORDER_DATE"
    .BaseItem = "5/16/89"
End With
```

Calculation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCalculationC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCalculationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCalculationA "}

Application object: Returns or sets the calculation mode. Can be one of the following **XlCalculation** constants: **xlCalculationAutomatic**, **xlCalculationManual**, or **xlCalculationSemiautomatic**.

PivotField object: Returns or sets the type of calculation done by the specified pivot field. Can be one of the following **XlPivotFieldCalculation** constants: **xlDifferenceFrom**, **xlIndex**, **xlNormal**, **xlPercentDifferenceFrom**, **xlPercentOf**, **xlPercentOfColumn**, **xlPercentOfRow**, **xlPercentOfTotal**, or **xlRunningTotal**. Valid only for data fields. Read/write **Long**.

Calculation Property Example

This example causes Microsoft Excel to calculate workbooks before they are saved to disk.

```
Application.Calculation = xlCalculateManual
Application.CalculateBeforeSave = True
```

This example sets the data field in the PivotTable on Sheet1 to calculate the difference from the base field, sets the base field to the field named "ORDER_DATE," and sets the base item to the item named "5/16/89."

```
With Worksheets("Sheet1").Range("A3")     .PivotField
    .Calculation = xlDifferenceFrom
    .BaseField = "ORDER_DATE"
    .BaseItem = "5/16/89"
End With
```

ChildField Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChildFieldC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChildFieldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChildFieldA "}

Returns a **PivotField** object that represents the child pivot field for the specified field (if the field is grouped and has a child field). Read-only.

Remarks

If the specified field has no child field, this property causes an error.

ChildField Property Example

This example displays the name of the child field for the field named "REGION2."

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
MsgBox "The name of the child field is " & _
   pvtTable.PivotFields("REGION2").ChildField.Name
```

ChildItems Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChildItemsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChildItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChildItemsA "}

Returns an object that represents either a single pivot item (a **PivotItem** object, Syntax 1) or a collection of all the pivot items (a **PivotItems** object, Syntax 2) that are group children in the specified field, or children of the specified item. Read-only.

Syntax 1

expression.**ChildItems(*Index*)**

Syntax 2

expression.**ChildItems**

expression Required. An expression that returns a **PivotField** or **PivotItem** object.

Index Optional **Variant**. The pivot item name or number (can be an array to specify more than one item).

ChildItems Property Example

This example adds the names of all the child items of the item named "vegetables" to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtItem In
pvtTable.PivotFields("product").PivotItems("vegetables").ChildItems
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtItem.Name
Next pvtItem
```

ColumnFields Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColumnFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColumnFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColumnFieldsA "}

Returns an object that represents either a single pivot field (a **PivotField** object, Syntax 1) or a collection of all the pivot fields (a **PivotFields** object, Syntax 2) that are currently shown as column fields. Read-only.

Syntax 1

expression.**ColumnFields(*Index*)**

Syntax 2

expression.**ColumnFields**

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

Index Optional **Variant**. The pivot field name or number (can be an array to specify more than one field).

ColumnFields Property Example

This example adds the field names of the PivotTable columns to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtField In pvtTable.ColumnFields
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtField.Name
Next pvtField
```

ColumnGrand Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColumnGrandC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColumnGrandX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColumnGrandA "}

True if the PivotTable shows grand totals for columns. Read/write **Boolean**.

ColumnGrand Property Example

This example sets the PivotTable to show grand totals for columns.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.ColumnGrand = True
```

ColumnRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColumnRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColumnRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColumnRangeA "}

Returns a **Range** object that represents the range that contains the PivotTable column area. Read-only.

ColumnRange Property Example

This example selects the column headers for the PivotTable.

```
Worksheets("Sheet1").Activate
Range("A3").Select
ActiveCell.PivotTable.ColumnRange.Select
```

CurrentPage Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCurrentPageC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCurrentPageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCurrentPageA "}

Returns or sets the current page showing for the page field (only valid for page fields). Read/write **String**.

Remarks

To set this property, set it to the name of the page. Set it to "All" to set all pages that are showing.

CurrentPage Property Example

This example sets the current page for the PivotTable on Sheet1 to the page named "Canada."

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.PivotFields("Country").CurrentPage = "Canada"
```

DataBodyRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataBodyRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataBodyRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataBodyRangeA "}

Returns a **Range** object that represents the range that contains the PivotTable data area. Read-only.

DataBodyRange Property Example

This example selects the active PivotTable data.

```
Worksheets("Sheet1").Activate
Range("A3").Select
ActiveCell.PivotTable.DataBodyRange.Select
```

DataFields Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataFieldsA "}

Returns an object that represents either a single pivot field (a **PivotField** object, Syntax 1) or a collection of all the pivot fields (a **PivotFields** object, Syntax 2) that are currently shown as data fields. Read-only.

Syntax 1

expression.**DataFields(*Index*)**

Syntax 2

expression.**DataFields**

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

Index Optional **Variant**. The pivot field name or number (can be an array to specify more than one field).

DataFields Property Example

This example adds the names for the PivotTable data fields to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtField In pvtTable.DataFields
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtField.Name
Next pvtField
```

DataLabelRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataLabelRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataLabelRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataLabelRangeA "}

Returns a **Range** object that represents the range that contains the labels for the PivotTable data fields. Read-only.

DataLabelRange Property Example

This example selects the data field labels in the PivotTable.

```
Worksheets("Sheet1").Activate
Range("A3").Select
ActiveCell.PivotTable.DataLabelRange.Select
```

DataRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataRangeA "}

Returns a **Range** object as shown in the following table. Read-only.

Object	Data range
Data field	Data contained in the field
Row, column, or page field	Items in the field
Item	Data qualified by the item

DataRange Property Example

This example selects the pivot items in the field named "REGION."

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
Worksheets("Sheet1").Activate
pvtTable.PivotFields("REGION").DataRange.Select
```

DataType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataTypeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataTypeA "}

PivotField object: Returns a constant describing the type of data in the pivot field. Can be one of the following **XlPivotFieldDataType** constants: **xlText**, **xlNumber**, or **xlDate**. Read-only **Long**.

Parameter object: Returns or sets the data type of the specified query parameter. Read/write **Long**.

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

xlParamTypeBigInt
xlParamTypeBinary
xlParamTypeBit
xlParamTypeChar
xlParamTypeDate
xlParamTypeDecimal
xlParamTypeDouble
xlParamTypeFloat
xlParamTypeInteger
xlParamTypeLongVarBinary
xlParamTypeLongVarChar
xlParamTypeNumeric
xlParamTypeReal
xlParamTypeSmallInt
xlParamTypeTime
xlParamTypeTimeStamp
xlParamTypeTinyInt
xlParamTypeUnknown
xlParamTypeVarBinary
xlParamTypeVarChar

DataType Property Example

This example displays the data type of the field named "ORDER_DATE."

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
Select Case pvtTable.PivotFields("ORDER_DATE").DataType
   Case Is = xlText
      MsgBox "The field contains text data"
   Case Is = xlNumber
      MsgBox "The field contains numeric data"
   Case Is = xlDate
      MsgBox "The field contains date data"
End Select
```

GroupLevel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGroupLevelC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGroupLevelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGroupLevelA "}

Returns the placement of the specified field within a group of fields (if the field is a member of a grouped set of fields). Read-only.

Remarks

The highest-level parent field (leftmost parent field) is level one, its child is level two, and so on.

GroupLevel Property Example

This example displays a message box if the field that contains the active cell is the highest-level parent field.

```
Worksheets("Sheet1").Activate
If ActiveCell.PivotField.GroupLevel = 1 Then
   MsgBox "This is the highest-level parent field."
End If
```

HasAutoFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasAutoFormatC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasAutoFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasAutoFormatA "}

True if the PivotTable is automatically formatted when it's refreshed or when fields are moved. Read/write **Boolean**.

HasAutoFormat Property Example

This example causes the PivotTable to be automatically reformatted when it's refreshed or when fields are moved.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.HasAutoFormat = True
```

HiddenFields Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHiddenFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHiddenFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHiddenFieldsA "}

Returns an object that represents either a single pivot field (a **PivotField** object, Syntax 1) or a collection of all the pivot fields (a **PivotFields** object, Syntax 2) that are currently not shown as row, column, page, or data fields. Read-only.

Syntax 1

expression.**HiddenFields(*Index*)**

Syntax 2

expression.**HiddenFields**

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

Index Optional **Variant**. The name or number of the pivot field to be returned (can be an array to specify more than one field).

HiddenFields Property Example

This example adds the hidden field names to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtField In pvtTable.HiddenFields
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtField.Name
Next pvtField
```

HiddenItems Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHiddenItemsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHiddenItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHiddenItemsA "}

Returns an object that represents either a single hidden pivot item (a **PivotItem** object, Syntax 1) or a collection of all the hidden pivot items (a **PivotItems** object, Syntax 2) in the specified field. Read-only.

Syntax 1

expression.**HiddenItems(*Index*)**

Syntax 2

expression.**HiddenItems**

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

Index Optional **Variant**. The number or name of the pivot item to be returned (can be an array to specify more than one item).

HiddenItems Property Example

This example adds the names of all the hidden items in the field named "product" to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtItem In pvtTable.PivotFields("product").HiddenItems
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtItem.Name
Next pvtItem
```

InnerDetail Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInnerDetailC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInnerDetailX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInnerDetailA "}

Returns or sets the name of the field that will be shown as detail when the **ShowDetail** property is **True** for the innermost row or column field. Read/write **String**.

InnerDetail Property Example

This example displays the name of the field that will be shown as detail when the **ShowDetail** property is **True** for the innermost row field or column field.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
MsgBox pvtTable.InnerDetail
```

LabelRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLabelRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLabelRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLabelRangeA "}

PivotField object: Returns a **Range** object that represents the cell (or cells) that contain the field label. Read-only.

PivotItem object: Returns a **Range** object that represents all the PivotTable cells that contain the item. Read-only.

LabelRange Property Example

This example selects the field button for the field named "ORDER_DATE."

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
Set pvtField = pvtTable.PivotFields("ORDER_DATE")
Worksheets("Sheet1").Activate
pvtField.LabelRange.Select
```

LocationInTable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLocationInTableC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLocationInTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLocationInTableA "}

Returns a constant that describes the part of the **PivotTable** that contains the upper-left corner of the specified range. Can be one of the following **XlLocationInTable** constants: **xlRowHeader**, **xlColumnHeader**, **xlPageHeader**, **xlDataHeader**, **xlRowItem**, **xlColumnItem**, **xlPageItem**, **xlDataItem**, or **xlTableBody**. Read-only **Long**.

LocationInTable Property Example

This example displays a message box that describes the location of the active cell within the PivotTable.

```
Worksheets("Sheet1").Activate
Select Case ActiveCell.LocationInTable
Case Is = xlRowHeader
   MsgBox "Active cell is part of a row header"
Case Is = xlColumnHeader
   MsgBox "Active cell is part of a column header"
Case Is = xlPageHeader
   MsgBox "Active cell is part of a page header"
Case Is = xlDataHeader
   MsgBox "Active cell is part of a data header"
Case Is = xlRowItem
   MsgBox "Active cell is part of a row item"
Case Is = xlColumnItem
   MsgBox "Active cell is part of a column item"
Case Is = xlPageItem
   MsgBox "Active cell is part of a page item"
Case Is = xlDataItem
   MsgBox "Active cell is part of a data item"
Case Is = xlTableBody
   MsgBox "Active cell is part of the table body"
End Select
```

PageFields Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageFieldsA "}

Returns an object that represents either a single pivot field (a **PivotField** object, Syntax 1) or a collection of all the pivot fields (a **PivotFields** object, Syntax 2) that are currently showing as page fields. Read-only.

Syntax 1

expression.**PageFields(*Index*)**

Syntax 2

expression.**PageFields**

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

Index Optional **Variant**. The name or number of the pivot field to be returned (can be an array to specify more than one field).

PageFields Property Example

This example adds the page field names to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtField In pvtTable.PageFields
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtField.Name
Next pvtField
```

PageRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageRangeA "}

Returns a **Range** object that represents the range that contains the PivotTable page area. Read-only.

PageRange Property Example

This example selects the PivotTable page headers.

```
Worksheets("Sheet1").Activate
Range("A3").Select
ActiveCell.PivotTable.PageRange.Select
```

ParentField Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproParentFieldC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproParentFieldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproParentFieldA "}

Returns a **PivotField** object that represents the pivot field that's the group parent of the object. The field must be grouped and have a parent field. Read-only.

ParentField Property Example

This example displays the name of the field that's the group parent of the field that contains the active cell.

```
Worksheets("Sheet1").Activate
MsgBox "The active field is a child of the field " & _
   ActiveCell.PivotField.ParentField.Name
```

ParentItem Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproParentItemC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproParentItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproParentItemA "}

Returns a **PivotItem** object that represents the parent pivot item in the parent **PivotField** object (the field must be grouped so that it has a parent). Read-only.

ParentItem Property Example

This example displays the name of the parent item for the item that contains the active cell.

```
Worksheets("Sheet1").Activate
MsgBox "This item is a subitem of " & _
   ActiveCell.PivotItem.ParentItem.Name
```

ParentItems Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproParentItemsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproParentItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproParentItemsA "}

Returns an object that represents either a single pivot item (a **PivotItem** object, Syntax 1) or a collection of all the pivot items (a **PivotItems** object, Syntax 2) that are group parents in the specified field. The specified field must be a group parent of another field. Read-only.

Syntax 1

expression.**ParentItems(*Index*)**

Syntax 2

expression.**ParentItems**

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

Index Optional **Variant**. The number or name of the pivot item to be returned (can be an array to specify more than one item).

ParentItems Property Example

This example creates a list containing the names of all the items that are group parents in the field named "product.".

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtItem In pvtTable.PivotFields("product").ParentItems
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtItem.Name
Next pvtItem
```

ParentShowDetail Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproParentShowDetailC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproParentShowDetailX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproParentShowDetailA "}

True if the specified item is showing because one of its parents is showing detail. **False** if the specified item isn't showing because one of its parents is hiding detail. This property is available only if the item is grouped. Read-only **Boolean**.

ParentShowDetail Property Example

This example displays a message if the pivot item that contains the active cell is visible because its parent item is showing detail.

```
Worksheets("Sheet1").Activate
Set pvtItem = ActiveCell.PivotItem
If pvtItem.ParentShowDetail = True Then
   MsgBox "Parent item is showing detail"
End If
```

PivotField Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPivotFieldC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPivotFieldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPivotFieldA "}

Returns a **PivotField** object that represents the pivot field containing the upper-left corner of the specified range. Read-only.

PivotField Property Example

This example displays the name of the pivot field that contains the active cell.

```
Worksheets("Sheet1").Activate
MsgBox "The active cell is in the field " & _
   ActiveCell.PivotField.Name
```

PivotFields Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotFieldsA "}

Returns an object that represents either a single pivot field (a **PivotField** object, Syntax 1) or a collection of both the visible and hidden pivot fields (a **PivotFields** object, Syntax 2) in the PivotTable. Read-only.

Syntax 1

expression.**PivotFields(*Index*)**

Syntax 2

expression.**PivotFields**

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

Index Optional **Variant**. The name or number of the pivot field to be returned (can be an array to specify more than one field).

PivotFields Method Example

This example adds the PivotTable field names to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtField In pvtTable.PivotFields
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtField.Name
Next pvtField
```

PivotItem Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPivotItemC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPivotItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPivotItemA "}

Returns a **PivotItem** object that represents the pivot item containing the upper-left corner of the specified range. Read-only.

PivotItem Property Example

This example displays the name of the pivot item that contains the active cell on Sheet1.

```
Worksheets("Sheet1").Activate
MsgBox "The active cell is in the item " & _
   ActiveCell.PivotItem.Name
```

PivotItems Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotItemsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotItemsA "}

Returns an object that represents either a single pivot item (a **PivotItem** object, Syntax 1) or a collection of all the visible and hidden pivot items (a **PivotItems** object, Syntax 2) in the specified field. Read-only.

Syntax 1

expression.**PivotItems(*Index*)**

Syntax 2

expression.**PivotItems**

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

Index Optional **Variant**. The name or number of the pivot item to be returned (can be an array to specify more than one item).

PivotItems Method Example

This example adds the names of all items in the field named "product" to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtitem In pvtTable.PivotFields("product").PivotItems
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtitem.Name
Next
```

PivotTable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPivotTableC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPivotTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPivotTableA "}

Returns a **PivotTable** object that represents the PivotTable containing the upper-left corner of the specified range. Read-only.

PivotTable Property Example

This example sets the current page for the PivotTable on Sheet1 to the page named "Canada."

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.PivotFields("Country").CurrentPage = "Canada"
```

PivotTables Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotTablesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotTablesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotTablesA "}

Returns an object that represents either a single PivotTable (a **PivotTable** object, Syntax 1) or a collection of all the PivotTables (a **PivotTables** object, Syntax 2) on a worksheet. Read-only.

Syntax 1

expression.**PivotTables(*Index*)**

Syntax 2

expression.**PivotTables**

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

Index Optional **Variant**. The name or number of the PivotTable (can be an array to specify more than one PivotTable).

PivotTables Method Example

This example sets the Sum of 1994 field in PivotTable1 to use the SUM function.

```
ActiveSheet.PivotTables("PivotTable1"). _
    PivotFields("Sum of 1994").Function = xlSum
```

PivotTableWizard Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotTableWizardC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotTableWizardX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotTableWizardA "}

Creates a **PivotTable**. This method doesn't display the PivotTable Wizard.

Syntax

expression.**PivotTableWizard(*SourceType, SourceData, TableDestination, TableName, RowGrand, ColumnGrand, SaveData, HasAutoFormat, AutoPage, Reserved, BackgroundQuery, OptimizeCache, PageFieldOrder, PageFieldWrapCount, ReadData, Connection*)**

expression Required. An expression that returns a **Worksheet** or **PivotTable** object.

SourceType Optional **Variant**. The source of the PivotTable data. Can be one of the following **XlPivotTableSourceType** constants.

Constant	Description
xlConsolidation	Multiple consolidation ranges
xlDatabase	Microsoft Excel list or database
xlExternal	Data from another application
xlPivotTable	Same source as another PivotTable

If you specify this argument, you must also specify ***SourceData***. If ***SourceType*** and ***SourceData*** omitted, Microsoft Excel assumes that the source type is **xlDatabase**, and the source data comes from the named range "Database." If this named range doesn't exist, Microsoft Excel uses the current region if the current selection is in a range of more than 10 cells that contain data. If this isn't true, this method will fail.

SourceData Optional **Variant**. The data for the new PivotTable. Can be a **Range** object, an array of ranges, or a text constant that represents the name of another PivotTable. For an external database, this is a two-element array. The first element is the connection string specifying the ODBC source for the data. The second element is the SQL query string used to get the data. If you specify this argument, you must specify ***SourceType***. If the active cell is inside the ***SourceData*** range, you must specify ***TableDestination***.

TableDestination Optional **Variant**. A **Range** object specifying where the PivotTable should be placed on the worksheet. If this argument is omitted, the PivotTable is placed at the active cell.

TableName Optional **Variant**. A string that specifies the name of the new PivotTable.

RowGrand Optional **Variant**. **True** to show grand totals for rows in the PivotTable. **False** to omit grand totals for rows.

ColumnGrand Optional **Variant**. **True** to show grand totals for columns in the PivotTable. **False** to omit grand totals for columns.

SaveData Optional **Variant**. **True** to save data with the PivotTable. **False** to save only the PivotTable definition.

HasAutoFormat Optional **Variant**. **True** to have Microsoft Excel automatically format the PivotTable when it's refreshed or when fields are moved.

AutoPage Optional **Variant**. Valid only if ***SourceType*** is **xlConsolidation**. **True** to have Microsoft Excel create a page field for the consolidation. If **False**, you must create the page field or fields.

Reserved Optional **Variant**. Not used by Microsoft Excel.

BackgroundQuery Optional **Variant**. **True** if queries for the PivotTable are performed asynchronously (in the background). The default value is **False**.

OptimizeCache Optional **Variant**. **True** to optimize the PivotTable cache when it's constructed. The default value is **False**.

PageFieldOrder Optional **Variant**. The order in which page fields are added to the PivotTable

layout. Can be one of the following **XlOrder** constants: **xlDownThenOver** or **xlOverThenDown**. The default value is **xlDownThenOver**.

PageFieldWrapCount Optional **Variant**. The number of PivotTable page fields in each column or row. The default value is 0 (zero).

ReadData Optional **Variant**. **True** to create a pivot cache containing all records from the external database; this cache may be very large. If **False**, some fields can be set to be server-based page fields before the data is actually read.

Connection Optional **Variant**. A string that contains ODBC settings that allow Microsoft Excel to connect to an ODBC data source; a URL that allows Microsoft Excel to connect to a Web data source; or a file that specifies a database or Web query. Overrides any previous setting of the **Connection** property of the **PivotCache** object.

PivotTableWizard Method Example

This example creates a new PivotTable from a Microsoft Excel database (contained in the range A1:C100).

```
ActiveSheet.PivotTableWizard xlDatabase, Range("A1:C100")
```

RefreshDate Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefreshDateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefreshDateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefreshDateA "}

Returns the date on which the PivotTable or pivot cache was last refreshed. Read-only **Date**.

RefreshDate Property Example

This example displays the date on which the PivotTable was last refreshed.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
dateString = Format(pvtTable.RefreshDate, "Long Date")
MsgBox "The data was last refreshed on " & dateString
```

RefreshName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefreshNameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefreshNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefreshNameA "}

Returns the name of the person who last refreshed the PivotTable data or pivot cache. Read-only **String**.

RefreshName Property Example

This example displays the name of the person who last refreshed the PivotTable.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
MsgBox "The data was last refreshed by " & pvtTable.RefreshName
```

RefreshTable Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRefreshTableC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRefreshTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRefreshTableA "}

Refreshes the PivotTable from the source data. Returns **True** if it's successful.

Syntax

expression.**RefreshTable**

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

RefreshTable Method Example

This example refreshes the PivotTable.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.RefreshTable
```

RowFields Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowFieldsA "}

Returns an object that represents either a single pivot field (a **PivotField** object, Syntax 1) or a collection of all the pivot fields (a **PivotFields** object, Syntax 2) that are currently showing as row fields. Read-only.

Syntax 1

expression.**RowFields(*Index*)**

Syntax 2

expression.**RowFields**

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

Index Optional **Variant**. The name or number of the pivot field to be returned (can be an array to specify more than one field).

RowFields Property Example

This example adds the PivotTable row field names to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtField In pvtTable.RowFields
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtField.Name
Next pvtField
```

RowGrand Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowGrandC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowGrandX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowGrandA "}

True if the PivotTable shows grand totals for rows. Read/write **Boolean**.

RowGrand Property Example

This example sets the PivotTable to show grand totals for rows.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.RowGrand = True
```

RowRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowRangeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowRangeA "}

Returns a **Range** object that represents the range including the PivotTable row area. Read-only.

RowRange Property Example

This example selects the PivotTable row headers.

```
Worksheets("Sheet1").Activate
Range("A3").Select
ActiveCell.PivotTable.RowRange.Select
```

SaveData Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSaveDataC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSaveDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSaveDataA "}

True if data for the PivotTable is saved with the workbook. **False** if only the PivotTable definition is saved. Read/write **Boolean**.

SaveData Property Example

This example sets the PivotTable to save data with the workbook.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.SaveData = True
```

ShowDetail Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowDetailC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowDetailX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowDetailA "}

True if the outline is expanded for the specified range (so that the detail of the column or row is visible). The specified range must be a single summary column or row in an outline. Read/write **Variant**.

For the **PivotItem** object (or the **Range** object if the range is in a PivotTable), this property is **True** if the pivot item is showing detail.

Remarks

If the specified range isn't in a PivotTable, the following remarks apply:

- The range must be in a single summary row or column.
- This property returns **False** if *any* of the children of the row or column are hidden.
- Setting this property to **True** is equivalent to unhiding all the children of the summary row or column.
- Setting this property to **False** is equivalent to hiding all the children of the summary row or column.

If the specified range is in a PivotTable, it's possible to set this property for more than one cell at a time if the range is contiguous. This property can be returned only if the range is a single cell.

ShowDetail Property Example

This example shows detail for the summary row of an outline on Sheet1. Before running this example, create a simple outline that contains a single summary row, and then collapse the outline so that only the summary row is showing. Select one of the cells in the summary row, and then run the example.

```
Worksheets("Sheet1").Activate
Set myRange = ActiveCell.CurrentRegion
lastRow = myRange.Rows.Count
myRange.Rows(lastRow).ShowDetail = True
```

ShowPages Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowPagesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowPagesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowPagesA "}

Creates a new PivotTable for each item in the page field. Each new PivotTable is created on a new worksheet.

Syntax

expression.**ShowPages(*PageField*)**

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

PageField Optional **Variant**. A string that names a single page field in the PivotTable.

ShowPages Method Example

This example creates a new PivotTable for each item in the page field, which is the field named "Country."

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
pvtTable.ShowPages "Country"
```

TableRange1 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTableRange1C "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTableRange1X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTableRange1A "}

Returns a **Range** object that represents the range containing the entire PivotTable, but doesn't include page fields. Read-only.

Remarks

The **TableRange2** property includes page fields.

TableRange1 Property Example

This example selects all of the PivotTable except its page fields.

```
Worksheets("Sheet1").Activate
Range("A3").PivotTable.TableRange1.Select
```

TableRange2 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTableRange2C "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTableRange2X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTableRange2A "}

Returns a **Range** object that represents the range containing the entire PivotTable, including page fields. Read-only.

Remarks

The **TableRange1** property doesn't include page fields.

TableRange2 Property Example

This example selects the entire PivotTable, including its page fields.

```
Worksheets("Sheet1").Activate
Range("A3").PivotTable.TableRange2.Select
```

TotalLevels Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTotalLevelsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTotalLevelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTotalLevelsA "}

Returns the total number of fields in the current field group. If the field is not grouped, **TotalLevels** returns the value 1. Read-only **Long**.

Remarks

All fields in a set of grouped fields have the same **TotalLevels** value.

TotalLevels Property Example

This example displays the total number of fields in the group that contains the active cell.

```
Worksheets("Sheet1").Activate
MsgBox "This group has " & _
   ActiveCell.PivotField.TotalLevels & " levels."
```

VisibleFields Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVisibleFieldsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVisibleFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVisibleFieldsA "}

Returns an object that represents either a single pivot field (a **PivotField** object, Syntax 1) or a collection of all the visible pivot fields (a **PivotFields** object, Syntax 2). Visible pivot fields are shown as row, column, page or data fields. Read-only.

Syntax 1

expression.**VisibleFields(*Index*)**

Syntax 2

expression.**VisibleFields**

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

Index Optional **Variant**. The name or number of the pivot field to be returned (can be an array to specify more than one field).

VisibleFields Property Example

This example adds the visible field names to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtField In pvtTable.VisibleFields
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtField.Name
Next pvtField
```

VisibleItems Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVisibleItemsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVisibleItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVisibleItemsA "}

Returns an object that represents either a single visible pivot item (a **PivotItem** object, Syntax 1) or a collection of all the visible pivot items (a **PivotItems** object, Syntax 2) in the specified field. Read-only.

Syntax 1

expression.**VisibleItems(*Index*)**

Syntax 2

expression.**VisibleItems**

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

Index Optional **Variant**. The number or name of the pivot item to be returned (can be an array to specify more than one item).

VisibleItems Property Example

This example adds the names of all visible items in the field named "Product" to a list on a new worksheet.

```
Set nwSheet = Worksheets.Add
nwSheet.Activate
Set pvtTable = Worksheets("Sheet2").Range("A1").PivotTable
rw = 0
For Each pvtItem In pvtTable.PivotFields("Product").VisibleItems
    rw = rw + 1
    nwSheet.Cells(rw, 1).Value = pvtItem.Name
Next
```

PageFieldStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageFieldStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageFieldStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageFieldStyleA"}

Returns or sets the style used in the bound page field area. The default value is a null string (no style is applied by default). Read/write **String**.

Remarks

This style is used as the default style for the background area, and it's applied before any user formatting. Cells vacated when a field is pivoted from the page field area to another location retain this style.

PageFieldStyle Property Example

This example sets the page field area of the PivotTable to the PurpleAndGold style.

```
Worksheets(1).PivotTables("Pivot1").PageFieldStyle = "PurpleAndGold"
```

TableStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTableStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTableStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTableStyleA"}

Returns or sets the style used in the PivotTable body. The default value is a null string (no style is applied by default). Read/write **String**.

Remarks

This style is used as the default style for the background area, and it's applied before any user formatting.

TableStyle Property Example

This example sets the body of the PivotTable to the PurpleAndGold style.

```
Worksheets(1).PivotTables("Pivot1").TableStyle = "PurpleAndGold"
```

VacatedStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVacatedStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVacatedStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVacatedStyleA"}

Returns or sets the style applied to cells vacated when the PivotTable is refreshed. The default value is a null string (no style is applied by default). Read/write **String**.

VacatedStyle Property Example

This example sets the vacated cells in the PivotTable to the BlackAndBlue style.

```
Worksheets(1).PivotTables("Pivot1").VacatedStyle = "BlackAndBlue"
```

PreserveFormatting Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPreserveFormattingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPreserveFormattingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPreserveFormattingA"}

True if PivotTable formatting is preserved when the PivotTable is refreshed or recalculated by operations such as pivoting, sorting, or changing page field items. Read/write **Boolean**.

PreserveFormatting Property Example

This example causes the PivotTable to preserve formatting.

```
Worksheets(1).PivotTables("Pivot1").PreserveFormatting = True
```

ManualUpdate Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproManualUpdateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproManualUpdateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproManualUpdateA"}

True if the PivotTable is recalculated only at the user's request. The default value is **False**. Read/write Boolean.

ManualUpdate Property Example

This example causes the PivotTable to recalculate only at the user's request and before the workbook is saved.

```
With Worksheets(1).PivotTables("Pivot1")
    .ManualUpdate = True
    .UpdateBeforeSave = True
End With
```

DisplayErrorString Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayErrorStringC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayErrorStringX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayErrorStringA"}

True if the PivotTable displays a custom error string in cells that contain errors. The default value is **False**. Read/write **Boolean**.

Remarks

Use the **ErrorString** property to set the custom error string.

This property is particularly useful for suppressing divide-by-zero errors when calculated fields are pivoted.

DisplayErrorString Property Example

This example causes the PivotTable to display a hyphen in cells that contain errors.

```
With Worksheets(1).PivotTables("Pivot1")
    .ErrorString = "-"
    .DisplayErrorString = True
End With
```

DisplayNullString Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayNullStringC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayNullStringX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayNullStringA"}

True if the PivotTable displays a custom string in cells that contain null values. The default value is **True**. Read/write **Boolean**.

Remarks

Use the **NullString** property to set the custom null string.

DisplayNullString Property Example

This example causes the PivotTable to display "NA" in cells that contain null values.

```
With Worksheets(1).PivotTables("Pivot1")
    .NullString = "NA"
    .DisplayNullString = True
End With
```

This example causes the PivotTable to display 0 (zero) in cells that contain null values.

```
Worksheets(1).PivotTables("Pivot1").DisplayNullString = False
```

ErrorString Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproErrorStringC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproErrorStringX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproErrorStringA"}

PivotTable object: Returns or sets the string displayed in cells that contain errors when the **DisplayErrorString** property is **True**. The default value is an empty string (""). Read/write **String**.

ODBCError object: Returns the ODBC error string. Read-only **String**.

ErrorString Property Example

This example causes the PivotTable to display a hyphen in cells that contain errors.

```
With Worksheets(1).PivotTables("Pivot1")
    .ErrorString = "-"
    .DisplayErrorString = True
End With
```

NullString Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNullStringC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNullStringX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNullStringA"}

Returns or sets the string displayed in cells that contain null values when the **DisplayNullString** property is **True**. The default value is an empty string (""). Read/write **String**.

NullString Property Example

This example causes the PivotTable to display "NA" in cells that contain null values.

```
With Worksheets(1).PivotTables("Pivot1")
    .NullString = "NA"
    .DisplayNullString = True
End With
```

ShowAllItems Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowAllItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowAllItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowAllItemsA"}

True if all items in the PivotTable are displayed, even if they don't contain summary data. The default value is **False**. Read/write **Boolean**.

ShowAllItems Property Example

This example causes the PivotTable to display all rows for the Month field, including months for which there's no data.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotFields("Month").ShowAllItems = True
```

SubtotalHiddenPageItems Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSubtotalHiddenPageItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSubtotalHiddenPageItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSubtotalHiddenPageItemsA"}

True if hidden page field items in the PivotTable are included in row and column subtotals, block totals, and grand totals. The default value is **False**. Read/write **Boolean**.

SubtotalHiddenPageItems Property Example

This example sets Pivot1 on worksheet one to exclude hidden page field items in subtotals.

```
Worksheets(1).PivotTables("Pivot1").SubtotalHiddenPageItems = True
```

PageFieldOrder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageFieldOrderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageFieldOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageFieldOrderA"}

Returns or sets the order in which page fields are added to the PivotTable layout. Can be one of the following **XlOrder** constants: **xlDownThenOver** or **xlOverThenDown**. The default constant is **xlDownThenOver**. Read/write **Long**.

PageFieldOrder Property Example

This example causes the PivotTable to draw three page fields in a row before starting a new row.

```
With Worksheets(1).PivotTables("Pivot1")
    .PageFieldOrder = xlOverThenDown
    .PageFieldWrapCount = 3
End With
```

PageFieldWrapCount Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageFieldWrapCountC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageFieldWrapCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageFieldWrapCountA"}

Returns or sets the number of PivotTable page fields in each column or row. Read/write **Long**.

PageFieldWrapCount Property Example

This example causes the PivotTable to draw three page fields in a row before starting a new row.

```
With Worksheets(1).PivotTables("Pivot1")
    .PageFieldOrder = xlOverThenDown
    .PageFieldWrapCount = 3
End With
```

CacheIndex Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCacheIndexC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCacheIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCacheIndexA"}

Returns or sets the index number of the PivotTable cache. Read/write **Long**.

Remarks

If you set the **CacheIndex** property so that one PivotTable uses the cache for a second PivotTable, the first PivotTable's fields must be a valid subset of the fields in the second PivotTable.

CacheIndex Property Example

This example sets the cache for the PivotTable named "Pivot1" to the cache of the PivotTable named "Pivot2."

```
Worksheets(1).PivotTables("Pivot1").CacheIndex = _
   Worksheets(1).PivotTables("Pivot2").CacheIndex
```

RecordCount Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRecordCountC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRecordCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRecordCountA"}

Returns the number of records in the PivotTable cache or the number of cache records that contain the specified item. Read-only **Long**.

RecordCount Property Example

This example displays the number of cache records that contain "Kiwi" in the "Products" field.

```
MsgBox Worksheets(1).PivotTables("Pivot1") _
    .PivotFields("Product").PivotItems("Kiwi").RecordCount
```

RefreshOnFileOpen Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefreshOnFileOpenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefreshOnFileOpenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefreshOnFileOpenA"}

True if the PivotTable cache or query table is automatically updated each time the workbook is opened. The default value is **False**. Read/write **Boolean**.

Remarks

Query tables and PivotTables are not automatically refreshed when you open the workbook by using the **Open** method in Visual Basic. Use the **Refresh** method to refresh the data after the workbook is open.

RefreshOnFileOpen Property Example

This example causes the PivotTable cache to automatically update each time the workbook is opened.

```
ActiveWorkbook.PivotCaches(1).RefreshOnFileOpen = True
```

EnableRefresh Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableRefreshC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableRefreshX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableRefreshA"}

True if the PivotTable cache or query table can be refreshed by the user. The default value is **True**. Read/write **Boolean**.

Remarks

The **RefreshOnFileOpen** property is ignored if the **EnableRefresh** property is set to **False**.

EnableRefresh Property Example

This example sets the PivotTable so that it cannot be refreshed.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotCache.EnableRefresh = False
```

BackgroundQuery Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBackgroundQueryC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBackgroundQueryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBackgroundQueryA"}

True if queries for the PivotTable or query table are performed asynchronously (in the background). Read/write **Boolean**.

BackgroundQuery Property Example

This example causes queries for the PivotTable to be performed in the background.

```
Worksheets(1).PivotTables("Pivot1").PivotCache.BackgroundQuery = True
```

OptimizeCache Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOptimizeCacheC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOptimizeCacheX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOptimizeCacheA"}

True if the PivotTable cache is optimized when it's constructed. The default value is **False**. Read/write **Boolean**.

Remarks

Cache optimization results in additional queries and degrades initial performance of the PivotTable.

OptimizeCache Property Example

This example causes the PivotTable cache to be optimized when it's constructed.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotCache.OptimizeCache = True
```

Tag Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTagC"}
{ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTagX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTagA"}

Returns or sets a string saved with the PivotTable. Read/write **String**.

Tag Property Example

This example sets the PivotTable **Tag** property.

```
Worksheets(1).PivotTables("Pivot1").Tag = "Product Sales by Region"
```

MemoryUsed Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMemoryUsedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMemoryUsedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMemoryUsedA"}

Application object: Returns the amount of memory that Microsoft Excel is currently using, in bytes. Read-only **Long**.

PivotCache or **PivotField** object: Returns the amount of memory currently being used by the object, in bytes. Read-only **Long**.

MemoryUsed Property Example

This example displays a message box showing the number of bytes that Microsoft Excel is currently using.

```
MsgBox "Microsoft Excel is currently using " & _
   Application.MemoryUsed & " bytes"
```

DragToColumn Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDragToColumnC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDragToColumnX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDragToColumnA"}

True if the pivot field can be dragged to the column position. The default value is **True**. Read/write **Boolean**.

DragToColumn Property Example

This example prevents the Year field from being dragged to the column position.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotFields("Year").DragToColumn = False
```

DragToHide Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDragToHideC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDragToHideX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDragToHideA"}

True if the field can be hidden by being dragged off the PivotTable. The default value is **True**. Read/write **Boolean**.

DragToHide Property Example

This example prevents the Year field from being dragged off the PivotTable.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotFields("Year").DragToHide = False
```

DragToPage Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDragToPageC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDragToPageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDragToPageA"}

True if the field can be dragged to the page position. The default value is **True**. Read/write **Boolean**.

DragToPage Property Example

This example prevents the Year field from being dragged to the page position.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotFields("Year").DragToPage = False
```

DragToRow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDragToRowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDragToRowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDragToRowA"}

True if the field can be dragged to the row position. The default value is **True**. Read/write **Boolean**.

DragToRow Property Example

This example prevents the Year field from being dragged to the row position.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotFields("Year").DragToRow = False
```

EnableDrilldown Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableDrilldownC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableDrilldownX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableDrilldownA"}

True if drilldown is enabled. The default value is **True**. Read/write **Boolean**.

Remarks

Setting this property for a PivotTable sets it for all fields in that PivotTable.

EnableDrilldown Property Example

This example disables drilldown for all fields in the PivotTable.

```
Worksheets(1).PivotTables("Pivot1").EnableDrilldown = False
```

EnableFieldDialog Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableFieldDialogC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableFieldDialogX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableFieldDialogA"}

True if the **PivotTable Field** dialog box is available when the user double-clicks the PivotTable field. The default value is **True**. Read/write **Boolean**.

Remarks

Setting this property for a PivotTable sets it for all fields in that PivotTable.

EnableFieldDialog Property Example

This example disables the **PivotTable Field** dialog box for the Year field.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotFields("Year").EnableFieldDialog = False
```

EnableWizard Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableWizardC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableWizardX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableWizardA"}

True if the PivotTable Wizard is available. The default value is **True**. Read/write **Boolean**.

EnableWizard Property Example

This example disables the PivotTable Wizard for the PivotTable named "Pivot1."

```
Worksheets(1).PivotTables("Pivot1").EnableWizard = False
```

IsCalculated Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIsCalculatedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIsCalculatedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIsCalculatedA"}

True if the pivot field or item is a calculated field or item. Read/write **Boolean**.

IsCalculated Property Example

This example disables the **PivotTable Field** dialog box if the PivotTable contains any calculated fields.

```
set pt = Worksheets(1).PivotTables("Pivot1")
For Each fld in pt.PivotFields
   If fld.IsCalculated Then pt.EnableFieldDialog = False
Next
```

PivotCaches Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotCachesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotCachesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotCachesA"}

Returns a **PivotCaches** collection that represents all the PivotTable caches in the specified workbook. Read-only.

Syntax

expression.**PivotCaches**

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

PivotCaches Method Example

This example causes the PivotTable cache to update automatically each time the workbook is opened.

```
ActiveWorkbook.PivotCaches(1).RefreshOnFileOpen = True
```

Refresh Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRefreshC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRefreshX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRefreshA"}

Updates the PivotTable cache or query table.

Syntax

expression.**Refresh(*BackgroundQuery*)**

expression Required. An expression that returns a **PivotCache** or **QueryTable** object.

BackgroundQuery Optional **Variant**. Used only with query tables. **True** to return control to the procedure as soon as a database connection is made and the the query is submitted (the query is updated in the background). **False** to return control to the procedure only after all data has been fetched to the worksheet. If this argument isn't specified, the setting of the **BackgroundQuery** property determines the query mode.

Remarks

The following remarks apply to the **QueryTable** object.

The **Refresh** method causes Microsoft Excel to connect to the query table's data source, execute the SQL query, and return data to the query table destination range. Until this method is called, the query table doesn't communicate with the data source.

When making the connection to the ODBC data source, Microsoft Excel uses the connection string specified by the **Connection** property. If the specified connection string is missing required values, the ODBC driver manager or the ODBC driver (or both) will display modal dialog boxes to prompt the user for the required information. If the **DisplayAlerts** property is **False**, dialog boxes aren't displayed and the **Refresh** method fails with the Insufficient Connection Information exception.

After Microsoft Excel makes a successful connection, it stores the completed connection string so that prompts won't be displayed for subsequent calls to the **Refresh** method during the same editing session. You can obtain the completed connection string by examining the value of the **Connection** property.

After the database connection is made, the SQL query is validated. If the query isn't valid, the **Refresh** method fails with the SQL Syntax Error exception.

If the query requires parameters, the **Parameters** collection must have been initialized with parameter binding information. If not enough parameters have been bound, the **Refresh** method fails with the Parameter Error exception. If parameters are set to prompt for their values, dialog boxes are displayed to the user regardless of the setting of the **DisplayAlerts** property. If the user cancels a parameter dialog box, the **Refresh** method halts and returns **False**. If there are extra parameters bound with the **Parameters** collection, the extra parameters are ignored.

The **Refresh** method returns **True** if the query is successfully completed or started; it returns **False** if the user cancels a connection or parameter dialog box.

To see whether the number of fetched rows exceeded the number of available rows on the worksheet, examine the **FetchedRowOverflow** property. This property is initialized every time the **Refresh** method is called.

Refresh Method Example

This example refreshes the PivotTable.

```
Worksheets(1).PivotTables(1).PivotCache.Refresh
```

SelectionMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSelectionModeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSelectionModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSelectionModeA"}

Returns or sets the PivotTable structured selection mode. Can be one of the following **XlPTSelectionMode** constants: **xlLabelOnly**, **xlDataAndLabel**, or **xlDataOnly**. Read/write **Long**.

SelectionMode Property Example

This example enables structured selection mode and then sets PivotTable one to allow only data to be selected.

```
Application.PivotTableSelection = True
Worksheets(1).PivotTables(1).SelectionMode = xlDataOnly
```

CalculatedFields Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCalculatedFieldsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCalculatedFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCalculatedFieldsA"}

Returns a **CalculatedFields** collection that represents all the calculated fields in the specified PivotTable. Read-only.

Syntax

expression.**CalculatedFields**

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

CalculatedFields Method Example

This example prevents the calculated fields from being dragged to the row position.

```
For Each fld in Worksheets(1).PivotTables("Pivot1").CalculatedFields
   fld.DragToRow = False
Next
```

CalculatedItems Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCalculatedItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCalculatedItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCalculatedItemsA"}

Returns a **CalculatedItems** collection that represents all the calculated items in the specified PivotTable. Read-only.

Syntax

expression.**CalculatedItems**

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

CalculatedItems Method Example

This example creates a list of calculated items and their formulas.

```
Set pt = Worksheets(1).PivotTables(1)
For Each ci In pt.PivotFields("Sales").CalculatedItems
   r = r + 1
   With Worksheets(2)
      .Cells(r, 1).Value = ci.Name
      .Cells(r, 2).Value = ci.Formula
   End With
Next
```

PivotCache Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotCacheC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotCacheX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotCacheA"}

Returns a **PivotCache** object that represents the cache for the specified PivotTable. Read-only.

Syntax

expression.**PivotCache**

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

PivotCache Method Example

This example causes the PivotTable cache to be optimized when it's constructed.

```
Worksheets(1).PivotTables("Pivot1") _
    .PivotCache.OptimizeCache = True
```

ShowChartTipNames Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowChartTipNamesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowChartTipNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowChartTipNamesA"}

True if charts show chart tip names. The default value is **True**. Read/write **Boolean**.

ShowChartTipNames Property Example

This example turns off chart tip names and values.

```
With Application
    .ShowChartTipNames = False
    .ShowChartTipValue = False
End With
```

ShowChartTipValues Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowChartTipValuesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowChartTipValuesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowChartTipValuesA"}

True if charts show chart tip values. The default value is **True**. Read/write **Boolean**.

ShowChartTipValues Property Example

This example turns off chart tip names and values.

```
With Application
    .ShowChartTipNames = False
    .ShowChartTipValue = False
End With
```

ShowWindow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowWindowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowWindowA"}

True if the embedded chart is displayed in a separate window. The **Chart** object used with this property must refer to an embedded chart. Read/write **Boolean**.

ShowWindow Property Example

This example causes the embedded chart to be displayed in a separate window.

```
Worksheets(1).ChartObjects(1).Chart.ShowWindow = True
```

AcceptAllChanges Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAcceptAllChangesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAcceptAllChangesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAcceptAllChangesA"}

Accepts all changes in the specified shared workbook.

Syntax

expression.**AcceptAllChanges**

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

AcceptAllChanges Method Example

This example accepts all changes in the active workbook.

```
ActiveWorkbook.AcceptAllChanges
```

RejectAllChanges Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRejectAllChangesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRejectAllChangesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRejectAllChangesA"}

Rejects all changes in the specified shared workbook.

Syntax

expression.**RejectAllChanges**

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

RejectAllChanges Method Example

This example rejects all changes in the active workbook.

```
ActiveWorkbook.RejectAllChanges
```

AcceptLabelsInFormulas Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAcceptLabelsInFormulasC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAcceptLabelsInFormulasX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAcceptLabelsInFormulasA"}

True if labels can be used in worksheet formulas. The default value is **True**. Read/write **Boolean**.

AcceptLabelsInFormulas Property Example

This example sets the **AcceptLabelsInFormulas** property for the active workbook and then sets cells B1:D1 on worksheet one to be column labels.

```
ActiveWorkbook.AcceptLabelsInFormulas = True
Worksheets(1).Range("b1:d1").FormulaLabel = xlColumnLabels
```

Add Method (Validation Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddValidationObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddValidationObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddValidationObjA"}

Adds data validation to the specified range.

Syntax

expression.**Add(*Type, AlertStyle, Operator, Formula1, Formula2*)**

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

Type Required **Long**. The validation type. Can be one of the following **XlDVType** constants: **xlValidateCustom**, **xlValidateDate**, **xlValidateDecimal**, **xlValidateInputOnly**, **xlValidateList**, **xlValidateTextLength**, **xlValidateTime**, or **xlValidateWholeNumber**.

AlertStyle Optional **Variant**. The validation alert style. Can be one of the following **XlDVAlertStyle** constants: **xlValidAlertInformation**, **xlValidAlertStop**, or **xlValidAlertWarning**.

Operator Optional **Variant**. The data validation operator. Can be one of the following **XlFormatConditionOperator** constants: **xlBetween**, **xlEqual**, **xlGreater**, **xlGreaterEqual**, **xlLess**, **xlLessEqual**, **xlNotBetween**, or **xlNotEqual**.

Formula1 Optional **Variant**. The first part of the data validation equation.

Formula2 Optional **Variant**. The second part of the data validation when ***Operator*** is **xlBetween** or **xlNotBetween** (otherwise, this argument is ignored).

Remarks

The **Add** method requires different arguments, depending on the validation type, as shown in the following table.

Validation type	Arguments
xlValidateCustom	***Formula1*** is required, ***Formula2*** is ignored. ***Formula1*** must contain an expression that evaluates to **True** when data entry is valid and **False** when data entry is invalid.
xlInputOnly	***AlertStyle***, ***Formula1***, or ***Formula2*** are used.
xlValidateList	***Formula1*** is required, ***Formula2*** is ignored. ***Formula1*** must contain either a comma-delimited list of values or a worksheet reference to this list.
xlValidateWholeNumber, **xlValidateDate**, **xlValidateDecimal**, **xlValidateTextLength**, or **xlValidateTime**	One of either ***Formula1*** or ***Formula2*** must be specified, or both may be specified.

Add Method (Validation Object) Example

This example adds data validation to cell E5.

```
With Range("e5").Validation
    .Add Type:=xlValidateWholeNumber, _
       AlertStyle:= xlValidAlertStop, _
       Operator:=xlBetween, Formula1:="5", Formula2:="10"
    .InputTitle = "Integers"
    .ErrorTitle = "Integers"
    .InputMessage = "Enter an integer from five to ten"
    .ErrorMessage = "You must enter a number from five to ten"
End With
```

AlertStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAlertStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAlertStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAlertStyleA"}

Returns the validation alert style. Can be one of the following **XlDVAlertStyle** constants: **xlValidAlertInformation**, **xlValidAlertStop**, or **xlValidAlertWarning**. Read-only **Long**.

Remarks

Use the **Add** method to set the alert style for a range. If the range already has data validation, use the **Modify** method to change the alert style.

AlertStyle Property Example

This example displays the alert style for cell E5.

```
MsgBox Range("e5").Validation.AlertStyle
```

AutoScaleFont Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoScaleFontC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoScaleFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoScaleFontA"}

True if the text in the object changes font size when the object size changes. The default value is **True**. Read/write **Variant**.

AutoScaleFont Property Example

This example adds a title to embedded chart one on the active worksheet, and it causes the title font to remain the same size whenever the chart size changes.

```
With ActiveSheet.ChartObjects(1).Chart
    .HasTitle = True
    .ChartTitle.Text = "1996 sales"
    .ChartTitle.AutoScaleFont = False
End With
```

BaseUnit Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBaseUnitC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBaseUnitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBaseUnitA"}

Returns or sets the base unit for the specified category axis. Can be one of the following **XlTimeUnit** constants: **xlDays**, **xlMonths**, or **xlYears**. Read/write **Long**.

Remarks

Setting this property has no visible effect if the **CategoryType** property for the specified axis is set to **xlCategoryScale**. The set value is retained, however, and takes effect when the **CategoryType** property is set to **xlTimeScale**.

You cannot set this property for a value axis.

BaseUnit Property Example

This example sets the category axis in embedded chart one on worksheet one to use a time scale, with months as the base unit.

```
With Worksheets(1).ChartObjects(1).Chart
   With .Axes(xlCategory)
      .CategoryType = xlTimeScale
      .BaseUnit = xlMonths
   End With
End With
```

BaseUnitIsAuto Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBaseUnitIsAutoC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBaseUnitIsAutoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBaseUnitIsAutoA"}

True if Microsoft Excel chooses appropriate base units for the specified category axis. The default value is **True**. Read/write **Boolean**.

Remarks

You cannot set this property for a value axis.

BaseUnitIsAuto Property Example

This example sets the category axis in embedded chart one on worksheet one to use a time scale with automatic base units.

```
With Worksheets(1).ChartObjects(1).Chart
   With .Axes(xlCategory)
      .CategoryType = xlTimeScale
      .BaseUnitIsAuto = True
   End With
End With
```

BubbleScale Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBubbleScaleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBubbleScaleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBubbleScaleA"}

Returns or sets the scale factor for bubbles in the specified chart group. Can be an integer value from 0 (zero) to 300, corresponding to a percentage of the default size. Applies only to bubble charts. Read/write **Long**.

BubbleScale Property Example

This example sets the bubble size in chart group one to 200% of the default size.

```
With Worksheets(1).ChartObjects(1).Chart
   .ChartGroups(1).BubbleScale = 200
End With
```

BubbleSizes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBubbleSizesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBubbleSizesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBubbleSizesA"}

Returns or sets a string in A1-style notation that refers to the worksheet cells containing the size data for the bubble chart. Applies only to bubble charts. Read/write **Variant**.

BubbleSizes Property Example

This example displays the cell reference for the cells that contain the bubble chart size data.

```
MsgBox Worksheets(1).ChartObjects(1).Chart _
    .SeriesCollection(1).BubbleSizes
```

Build Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBuildC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBuildX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBuildA"}

Returns the Microsoft Excel build number. Read-only **Long**.

Remarks

It's usually safer to test the **Version** property, unless you're sure you need to know the build number.

Build Property Example

This example tests the **Build** property.

```
If Application.Build > 2500 Then
   ' build-dependent code here
End If
```

CategoryType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCategoryTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCategoryTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCategoryTypeA"}

Returns or sets the category axis type. Can be one of the following **XlCategoryType** constants: **xlCategoryScale**, **xlTimeScale**, or **xlAutomaticScale**. Read/write **Long**.

Remarks

You cannot set this property for a value axis.

CategoryType Property Example

This example sets the category axis in embedded chart one on worksheet one to use a time scale, with months as the base unit.

```
With Worksheets(1).ChartObjects(1).Chart
   With .Axes(xlCategory)
      .CategoryType = xlTimeScale
      .BaseUnit = xlMonths
   End With
End With
```

ChartType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChartTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChartTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChartTypeA"}

Returns or sets the chart type. Read/write **Long**.

Can be one of the following **XlChartType** constants.

Chart type	Description	Constant
Column	Clustered Column	**xlColumnClustered**
	3D Clustered Column	**xl3DColumnClustered**
	Stacked Column	**xlColumnStacked**
	3D Stacked Column	**xl3DColumnStacked**
	100% Stacked Column	**xlColumnStacked100**
	3D 100% Stacked Column	**xl3DColumnStacked100**
	3D Column	**xl3DColumn**
Bar	Clustered Bar	**xlBarClustered**
	3D Clustered Bar	**xl3DBarClustered**
	Stacked Bar	**xlBarStacked**
	3D Stacked Bar	**xl3DBarStacked**
	100% Stacked Bar	**xlBarStacked100**
	3D 100% Stacked Bar	**xl3DBarStacked100**
Line	Line	**xlLine**
	Line with Markers	**xlLineMarkers**
	Stacked Line	**xlLineStacked**
	Stacked Line with Markers	**xlLineMarkersStacked**
	100% Stacked Line	**xlLineStacked100**
	100% Stacked Line with Markers	**xlLIneMarkersStacked100**
	3D Line	**xl3DLine**
Pie	Pie	**xlPie**
	Exploded Pie	**xlPieExploded**
	3D Pie	**xl3Dpie**
	Exploded 3D Pie	**xl3DPieExploded**
	Pie of Pie	**xlPieOfPie**
	Bar of Pie	**xlBarOfPie**
XY (Scatter)	Scatter	**xlXYScatter**
	Scatter with Smoothed Lines	**xlXYScatterSmooth**
	Scatter with Smoothed Lines and No Data Markers	**xlXYScatterSmoothNoMarkers**
	Scatter with Lines	**xlXYScatterLines**
	Scatter with Lines and No Data Markers	**xlXYScatterLinesNoMarkers**
Bubble	Bubble	**xlBubble**
	Bubble with 3D effects	**xlBubble3DEffect**
Area	Area	**xlArea**
	3D Area	**xl3DArea**
	Stacked Area	**xlAreaStacked**

	3D Stacked Area	**xl3DAreaStacked**
	100% Stacked Area	**xlAreaStacked100**
	3D 100% Stacked Area	**xl3DAreaStacked100**
Doughnut	Doughnut	**xlDoughnut**
	Exploded Doughnut	**xlDoughnutExploded**
Radar	Radar	**xlRadar**
	Radar with Data Markers	**xlRadarMarkers**
	Filled Radar	**xlRadarFilled**
Surface	3D Surface	**xlSurface**
	Surface (Top View)	**xlSurfaceTopView**
	3D Surface (wireframe)	**xlSurfaceWireframe**
	Surface (Top View wireframe)	**xlSurfaceTopViewWireframe**
Stock Quotes	High-Low-Close	**xlStockHLC**
	Volume-High-Low-Close	**xlStockVHLC**
	Open-High-Low-Close	**xlStockOHLC**
	Volume-Open-High-Low-Close	**xlStockVOHLC**
Cylinder	Clustered Cylinder Column	**xlCylinderColClustered**
	Clustered Cylinder Bar	**xlCylinderBarClustered**
	Stacked Cylinder Column	**xlCylinderColStacked**
	Stacked Cylinder Bar	**xlCylinderBarStacked**
	100% Stacked Cylinder Column	**xlCylinderColStacked100**
	100% Stacked Cylinder Bar	**xlCylinderBarStacked100**
	3D Cylinder Column	**xlCylinderCol**
Cone	Clustered Cone Column	**xlConeColClustered**
	Clustered Cone Bar	**xlConeBarClustered**
	Stacked Cone Column	**xlConeColStacked**
	Stacked Cone Bar	**xlConeBarStacked**
	100% Stacked Cone Column	**xlConeColStacked100**
	100% Stacked Cone Bar	**xlConeBarStacked100**
	3D Cone Column	**xlConeCol**
Pyramid	Clustered Pyramid Column	**xlPyramidColClustered**
	Clustered Pyramid Bar	**xlPyramidBarClustered**
	Stacked Pyramid Column	**xlPyramidColStacked**
	Stacked Pyramid Bar	**xlPyramidBarStacked**
	100% Stacked Pyramid Column	**xlPyramidColStacked100**
	100% Stacked Pyramid Bar	**xlPyramidBarStacked100**
	3D Pyramid Column	**xlPyramidCol**

ChartType Property Example

This example sets the bubble size in chart group one to 200% of the default size if the chart is a 2D bubble chart.

```
With Worksheets(1).ChartObjects(1).Chart
   If .ChartType = xlBubble Then
      .ChartGroups(1).BubbleScale = 200
   End If
End With
```

CircleInvalid Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCircleInvalidC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCircleInvalidX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCircleInvalidA"}

Circles invalid entries on the worksheet.

Syntax

expression.**CircleInvalid**

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

CircleInvalid Method Example

This example circles invalid entries on worksheet one.

```
Worksheets(1).CircleInvalid
```

ClearCircles Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearCirclesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearCirclesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearCirclesA"}

Clears circles from invalid entries on the worksheet.

Syntax

expression.**ClearCircles**

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

Remarks

Use the **CircleInvalid** method to circle cells that contain invalid data.

ClearCircles Method Example

This example clears circles from invalid entries on worksheet one.

```
Worksheets(1).ClearCircles
```

DisplayPageBreaks Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayPageBreaksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayPageBreaksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayPageBreaksA"}

True if page breaks (both automatic and manual) on the specified worksheet are displayed. Read/write **Boolean**.

Remarks

You can't set this property if you don't have a printer installed.

DisplayPageBreaks Property Example

This example causes Sheet1 to display page breaks.

```
Worksheets("Sheet1").DisplayPageBreaks = True
```

ErrorMessage Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproErrorMessageC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproErrorMessageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproErrorMessageA"}

Returns or sets the data validation error message. Read/write **String**.

ErrorMessage Property Example

This example adds data validation to cell E5 and specifies both the input and error messages.

```
With Range("e5").Validation
    .Add Type:=xlValidateWholeNumber, _
       AlertStyle:= xlValidAlertStop, _
       Operator:=xlBetween, Formula1:="5", Formula2:="10"
    .InputTitle = "Integers"
    .ErrorTitle = "Integers"
    .InputMessage = "Enter an integer from five to ten"
    .ErrorMessage = "You must enter a number from five to ten"
End With
```

FormulaLabel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormulaLabelC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormulaLabelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormulaLabelA"}

Returns or sets the formula label type for the specified range. Can be **xlNone** if the range contains no labels, or one of the following **XlFormulaLabel** constants: **xlRowLabels**, **xlColumnLabels**, or **xlMixedLabels**. Read/write **Variant**.

FormulaLabel Property Example

This example topic sets the **AcceptLabelsInFormulas** property and then sets cells B1:D1 to be column labels.

```
ActiveWorkbook.AcceptLabelsInFormulas = True
Worksheets(1).Range("b1:d1").FormulaLabel = xlColumnLabels
```

Has3DEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHas3DEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHas3DEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHas3DEffectA"}

True if the series has a three-dimensional appearance. Applies only to bubble charts. Read/write **Boolean**.

Has3DEffect Property Example

This example gives series one on the embedded bubble chart a three-dimensional appearance.

```
With Worksheets(1).ChartObjects(1).Chart
    .SeriesCollection(1).Has3DEffect = True
End With
```

InputMessage Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInputMessageC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInputMessageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInputMessageA"}

Returns or sets the data validation input message. Read/write **String**.

InputMessage Property Example

This example adds data validation to cell E5 and specifies both the input and error messages.

```
With Range("e5").Validation
   .Add Type:=xlValidateWholeNumber, _
      AlertStyle:= xlValidAlertStop, _
      Operator:=xlBetween, Formula1:="5", Formula2:="10"
   .InputTitle = "Integers"
   .ErrorTitle = "Integers"
   .InputMessage = "Enter an integer from five to ten"
   .ErrorMessage = "You must enter a number from five to ten"
End With
```

InsideHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInsideHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInsideHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInsideHeightA"}

Returns the inside height of the plot area, in points. Read-only **Double**.

Remarks

The plot area used for this measurement doesn't include the axis labels. The **Height** property for the plot area uses the bounding rectangle that includes the axis labels.

InsideHeight Property Example

This example draws a dotted rectangle around the inside of the plot area in Chart1.

```
With Charts("chart1")
   Set pa = .PlotArea
   With .Shapes.AddShape(msoShapeRectangle, _
         pa.InsideLeft, pa.InsideTop, pa.InsideWidth, pa.InsideHeight)
      .Fill.Transparency = 1
      .Line.DashStyle = msoLineDashDot
   End With
End With
```

InsideLeft Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInsideLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInsideLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInsideLeftA"}

Returns the distance from the chart edge to the inside left edge of the plot area, in points. Read-only **Double**.

Remarks

The plot area used for this measurement doesn't include the axis labels. The **Left** property for the plot area uses the bounding rectangle that includes the axis labels.

InsideLeft Property Example

This example draws a dotted rectangle around the inside of the plot area in Chart1.

```
With Charts("chart1")
   Set pa = .PlotArea
   With .Shapes.AddShape(msoShapeRectangle, _
         pa.InsideLeft, pa.InsideTop, pa.InsideWidth, pa.InsideHeight)
      .Fill.Transparency = 1
      .Line.DashStyle = msoLineDashDot
   End With
End With
```

InsideTop Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInsideTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInsideTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInsideTopA"}

Returns the distance from the chart edge to the inside top edge of the plot area, in points. Read-only **Double**.

Remarks

The plot area used for this measurement doesn't include the axis labels. The **Top** property for the plot area uses the bounding rectangle that includes the axis labels.

InsideTop Property Example

This example draws a dotted rectangle around the inside of the plot area in Chart1.

```
With Charts("chart1")
   Set pa = .PlotArea
   With .Shapes.AddShape(msoShapeRectangle, _
         pa.InsideLeft, pa.InsideTop, pa.InsideWidth, pa.InsideHeight)
      .Fill.Transparency = 1
      .Line.DashStyle = msoLineDashDot
   End With
End With
```

InsideWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInsideWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInsideWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInsideWidthA"}

Returns the inside width of the plot area, in points. Read-only **Double**.

Remarks

The plot area used for this measurement doesn't include the axis labels. The **Width** property for the plot area uses the bounding rectangle that includes the axis labels.

InsideWidth Property Example

This example draws a dotted rectangle around the inside of the plot area in Chart1.

```
With Charts("chart1")
   Set pa = .PlotArea
   With .Shapes.AddShape(msoShapeRectangle, _
         pa.InsideLeft, pa.InsideTop, pa.InsideWidth, pa.InsideHeight)
      .Fill.Transparency = 1
      .Line.DashStyle = msoLineDashDot
   End With
End With
```

PlotBy Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPlotByC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPlotByX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPlotByA"}

Returns or sets the way columns or rows are used as data series on the chart. Can be one of the following **XlRowCol** constants: **xlColumns** or **xlRows**. Read/write **Long**.

PlotBy Property Example

This example causes the embedded chart to plot data by columns.

```
Worksheets(1).ChartObjects(1).Chart.PlotBy = xlColumns
```

ProtectChartObject Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectChartObjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectChartObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectChartObjectA"}

True if the embedded chart frame cannot be moved, resized, or deleted. Read/write **Boolean**.

ProtectChartObject Property Example

This example protects embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).ProtectChartObject = True
```

ProtectData Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectDataC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectDataA"}

True if series formulas cannot be modified by the user. Read/write **Boolean**.

ProtectData Property Example

This example protects the data on embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Chart.ProtectData = True
```

ProtectFormatting Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectFormattingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectFormattingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectFormattingA"}

True if chart formatting cannot be modified by the user. Read/write **Boolean**.

Remarks

When this property is **True**, the **Object** command on the **Format** menu is disabled and chart elements cannot be added, moved, resized, or deleted.

ProtectFormatting Property Example

This example protects the formatting of embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Chart.ProtectFormatting = True
```

ProtectGoalSeek Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectGoalSeekC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectGoalSeekX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectGoalSeekA"}

True if the user cannot modify chart data points with mouse actions. Read/write **Boolean**.

ProtectGoalSeek Property Example

This example protects the data points on embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Chart.ProtectGoalSeek = True
```

ProtectSelection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectSelectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectSelectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectSelectionA"}

True if chart elements cannot be selected. Read/write **Boolean**.

Remarks

When this property is **True**, shapes cannot be added to the chart, and the Click and DoubleClick events for chart elements don't occur.

ProtectSelection Property Example

This example prevents chart elements from being selected on embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Chart.ProtectSelection = True
```

ShrinkToFit Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShrinkToFitC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShrinkToFitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShrinkToFitA"}

True if text automatically shrinks to fit in the available column width. Returns **Null** if this property isn't set to the same value for all cells in the specified range. Read/write **Variant**.

ShrinkToFit Property Example

This example causes text in row one to automatically shrink to fit in the available column width.

```
Rows(1).ShrinkToFit = True
```

SplitValue Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSplitValueC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSplitValueX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSplitValueA"}

Returns or sets the threshold value separating the two sections of either a pie of pie chart or a bar of pie chart. Read/write **Variant**.

SplitValue Property Example

This example must be run on either a pie of pie chart or a bar of pie chart. The example splits the two sections of the chart by value, combining all values under 10 in the primary pie and displaying them in the secondary section.

```
With Worksheets(1).ChartObjects(1).Chart.ChartGroups(1)
    .SplitType = xlSplitByValue
    .SplitValue = 10
    .VaryByCategories = True
End With
```

SplitType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSplitTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSplitTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSplitTypeA"}

Returns or sets the way the two sections of either a pie of pie chart or a bar of pie chart are split. Can be one of the following **XlChartSplitType** constants: **xlSplitByPosition**, **xlSplitByPercentValue**, **xlSplitByCustomSplit**, or **xlSplitByValue**. Read/write **Long**.

SplitType Property Example

This example must be run on either a pie of pie chart or a bar of pie chart. The example splits the two sections of the chart by value, combining all values under 10 in the primary pie and displaying them in the secondary section.

```
With Worksheets(1).ChartObjects(1).Chart.ChartGroups(1)
    .SplitType = xlSplitByValue
    .SplitValue = 10
    .VaryByCategories = True
End With
```

SecondPlotSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSecondPlotSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSecondPlotSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSecondPlotSizeA"}

Returns or sets the size of the secondary section of either a pie of pie chart or a bar of pie chart, as a percentage of the size of the primary pie. Can be a value from 5 to 200. Read/write **Long**.

SecondPlotSize Property Example

This example must be run on either a pie of pie chart or a bar of pie chart. The example splits the two sections of the chart by value, combining all values under 10 in the primary pie and displaying them in the secondary section. The secondary section is 50 percent of the size of the primary pie.

```
With Worksheets(1).ChartObjects(1).Chart.ChartGroups(1)
    .SplitType = xlSplitByValue
    .SplitValue = 10
    .VaryByCategories = True
    .SecondPlotSize = 50
End With
```

SecondaryPlot Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSecondaryPlotC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSecondaryPlotX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSecondaryPlotA"}

True if the point is in the secondary section of either a pie of pie chart or a bar of pie chart. Applies only to points on pie of pie charts or bar of pie charts. Read/write **Boolean**.

SecondaryPlot Property Example

This example must be run on either a pie of pie chart or a bar of pie chart. The example moves point four to the secondary section of the chart.

```
With Worksheets(1).ChartObjects(1).Chart.SeriesCollection(1)
   .Points(4).SecondaryPlot = True
End With
```

Validation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproValidationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproValidationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproValidationA"}

Returns the **Validation** object that represents data validation for the specified range. Read-only.

Validation Property Example

This example causes data validation for cell E5 to allow blank values.

```
Range("e5").Validation.IgnoreBlank = True
```

WorksheetFunction Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWorksheetFunctionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWorksheetFunctionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWorksheetFunctionA"}

Returns the **WorksheetFunction** object. Read-only.

WorksheetFunction Property Example

This example displays the result of applying the **Min** worksheet function to the range A1:A10.

```
Set myRange = Worksheets("Sheet1").Range("A1:C10")
answer = Application.WorksheetFunction.Min(myRange)
MsgBox answer
```

IgnoreBlank Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIgnoreBlankC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIgnoreBlankX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIgnoreBlankA"}

True if blank values are permitted by the range data validation. Read/write **Boolean**.

Remarks

If the **IgnoreBlank** property is **True**, cell data is considered valid if the cell is blank, or if a cell referenced by either the **MinVal** or **MaxVal** property is blank.

IgnoreBlank Property Example

This example causes data validation for cell E5 to allow blank values.

```
Range("e5").Validation.IgnoreBlank = True
```

InCellDropdown Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInCellDropdownC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInCellDropdownX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInCellDropdownA"}

True if data validation displays a drop-down list that contains acceptable values. Read/write **Boolean**.

Remarks

This property is ignored if the validation type isn't **xlValidateList**.

Use the ***Minimum*** argument of the **Add** or **Modify** method of the **Validation** object to specify the range that contains valid data.

InCellDropdown Property Example

This example adds data validation to cell E5. The range A1:A10 contains the acceptable values for the cell and the cell displays a drop-down list that contains those values.

```
With Range("e5").Validation
    .Add xlValidateList, xlValidAlertStop, xlBetween, "=$A$1:$A$10"
    .InCellDropdown = True
End With
```

Modify Method (Validation Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthModifyValidationObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthModifyValidationObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthModifyValidationObjA"}

Modifies data validation for a range.

Syntax

expression.**Modify**(***Type, AlertStyle, Minimum, Maximum***)

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

Type Required **Long**. The validation type. Can be one of the following **XlDVType** constants: **xlValidateCustom**, **xlValidateDate**, **xlValidateDecimal**, **xlValidateInputOnly**, **xlValidateList**, **xlValidateTextLength**, **xlValidateTime**, or **xlValidateWholeNumber**.

AlertStyle Optional **Variant**. The validation alert style. Can be one of the following **XlDVAlertStyle** constants: **xlValidAlertInformation**, **xlValidAlertStop**, or **xlValidAlertWarning**.

Operator Optional **Variant**. The data validation operator. Can be one of the following **XlFormatConditionOperator** constants: **xlBetween**, **xlEqual**, **xlGreater**, **xlGreaterEqual**, **xlLess**, **xlLessEqual**, **xlNotBetween**, or **xlNotEqual**.

Formula1 Optional **Variant**. The first part of the data validation equation.

Formula2 Optional **Variant**. The second part of the data validation when ***Operator*** is **xlBetween** or **xlNotBetween** (otherwise, this argument is ignored).

Remarks

The **Modify** method requires different arguments, depending on the validation type, as shown in the following table.

Validation type	Arguments
xlValidateCustom	***Formula1*** is required; ***Formula2*** is ignored. ***Formula1*** must contain an expression that evaluates to **True** when data entry is valid and **False** when data entry is invalid.
xlInputOnly	***AlertStyle***, ***Formula1***, and ***Formula2*** are not used.
xlValidateList	***Formula1*** is required; ***Formula2*** is ignored. ***Formula1*** must contain either a comma-delimited list of values or a worksheet reference to the list.
xlValidateWholeNumber, **xlValidateDate**, **xlValidateDecimal**, **xlValidateTextLength**, or **xlValidateTime**	***Formula1*** or ***Formula2***, or both, must be specified.

Modify Method (Validation Object) Example

This example changes data validation for cell E5.

```
Range("e5").Validation _
    .Modify xlValidateList, xlValidAlertStop, xlBetween, "=$A$1:$A$10"
```

Name Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNameA "}

Returns or sets the name of the object. The name of a **Range** object is a **Name** object. For every other type of object, the name is a string.

Name Property Example

This example displays the name of style one in the active workbook, first in the language of the macro and then in the language of the user.

```
With ActiveWorkbook.Styles(1)
   MsgBox "The name of the style is " & .Name
   MsgBox "The localized name of the style is " & .NameLocal
End With
```

ActiveSheet Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproActiveSheetC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproActiveSheetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproActiveSheetA "}

Returns an object that represents the active sheet (the sheet on top) in the active workbook or in the specified window or workbook. Returns **Nothing** if no sheet is active. Read-only.

Remarks

If you don't specify an object qualifier, this property returns the active sheet in the active workbook.

If a workbook appears in more than one window, the **ActiveSheet** property may be different in different windows.

ActiveSheet Property Example

This example displays the name of the active sheet.

```
MsgBox "The name of the active sheet is " & ActiveSheet.Name
```

Address Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAddressC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAddressX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAddressA "}

Hyperlink object (Syntax 1): Returns or sets the address of the target document. Read/write **String**.

Range object (Syntax 2): Returns the range reference in the language of the macro. Read-only **String**.

Syntax 1

expression.**Address**

Syntax 2

expression.**Address(*RowAbsolute, ColumnAbsolute, ReferenceStyle, External, RelativeTo*)**

expression Required. An expression that returns a **Hyperlink** object (Syntax 1) or a **Range** object (Syntax 2).

RowAbsolute Optional **Variant**. **True** to return the row part of the reference as an absolute reference. The default value is **True**.

ColumnAbsolute Optional **Variant**. **True** to return the column part of the reference as an absolute reference. The default value is **True**.

ReferenceStyle Optional **Variant**. Can be one of the following **XlReferenceStyle** constants: **xlA1** or **xlR1C1**. Use **xlA1** to return an A1-style reference. Use **xlR1C1** to return an R1C1-style reference. The default value is **xlA1**

External Optional **Variant**. **True** to return an external reference. **False** to return a local reference. The default value is **False**.

RelativeTo Optional **Variant**. If ***RowAbsolute*** and ***ColumnAbsolute*** are **False**, and ***ReferenceStyle*** is **xlR1C1**, you must include a starting point for the relative reference. This argument is a **Range** object that defines the starting point.

Remarks

If the reference contains more than one cell, ***RowAbsolute*** and ***ColumnAbsolute*** apply to all rows and columns.

Address Property Example

The following example displays four different representations of the same cell address on Sheet1. The comments in the example are the addresses that will be displayed in the message boxes.

```
Set mc = Worksheets("Sheet1").Cells(1, 1)
MsgBox mc.Address()                                 ' $A$1
MsgBox mc.Address(RowAbsolute:=False)               ' $A1
MsgBox mc.Address(ReferenceStyle:=xlR1C1)           ' R1C1
MsgBox mc.Address(ReferenceStyle:=xlR1C1, _
   RowAbsolute:=False,     _
   ColumnAbsolute:=False,  _
   RelativeTo:=Worksheets(1).Cells(3, 3))           ' R[-2]C[-2]
```

AddressLocal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAddressLocalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAddressLocalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAddressLocalA "}

Returns the range reference in the language of the user. Read-only **String**.

Syntax

expression.**AddressLocal(*RowAbsolute, ColumnAbsolute, ReferenceStyle, External, RelativeTo*)**

RowAbsolute Optional **Variant**. **True** to return the row part of the reference as an absolute reference. The default value is **True**.

ColumnAbsolute Optional **Variant**. **True** to return the column part of the reference as an absolute reference. The default value is **True**.

ReferenceStyle Optional **Variant**. Can be one of the following **XlReferenceStyle** constants: **xlA1** or **xlR1C1**. Use **xlA1** to return an A1-style reference. Use **xlR1C1** to return an R1C1-style reference. The default value is **xlA1**

External Optional **Variant**. **True** to return an external reference. **False** to return a local reference. The default value is **False**.

RelativeTo Optional **Variant**. If ***RowAbsolute*** and ***ColumnAbsolute*** are **False**, and ***ReferenceStyle*** is **xlR1C1**, you must include a starting point for the relative reference. This argument is a **Range** object that defines the starting point.

Remarks

If the reference contains more than one cell, ***RowAbsolute*** and ***ColumnAbsolute*** apply to all rows and columns.

AddressLocal Property Example

Assume that the following example was created in the American English version of Microsoft Excel and was then run in the German version. The example displays the text shown in the comments.

```
Set mc = Worksheets(1).Cells(1, 1)
MsgBox mc.AddressLocal()                          ' $A$1
MsgBox mc.AddressLocal(RowAbsolute:=False)        ' $A1
MsgBox mc.AddressLocal(ReferenceStyle:=xlR1C1)    ' Z1S1
MsgBox mc.AddressLocal(ReferenceStyle:=xlR1C1, _
    RowAbsolute:=False, _
    ColumnAbsolute:=False, _
    RelativeTo:=Worksheets(1).Cells(3, 3))        ' Z(-2)S(-2)
```

Author Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAuthorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAuthorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAuthorA "}

Returns or sets the author of the comment. Read-only **String**.

Author Property Example

This example deletes all comments added by Jean Selva on the active sheet.

```
For Each c in ActiveSheet.Comments
   If c.Author = "Jean Selva" Then c.Delete
Next
```

Caller Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCallerC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCallerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCallerA "}

Returns information about how Visual Basic was called (for more information, see the Remarks section).

Syntax

expression.**Caller(*Index*)**

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

Index Optional **Variant**. An index to the array. This argument is used only when the property returns an array (for more information, see the Remarks section).

Remarks

This property returns information about how Visual Basic was called, as shown in the following table.

Caller	Return value
A custom function entered in a single cell	A **Range** object specifying that cell
A custom function that is part of an array formula in a range of cells	A **Range** object specifying that range of cells
An Auto_Open, Auto_Close, Auto_Activate, or Auto_Deactivate macro	The name of the document as text
A macro set by either the **OnDoubleClick** or **OnEntry** property	The name of the chart object identifier or cell reference (if applicable) to which the macro applies
The **Macro** dialog box (**Tools** menu), or any caller not described above	The #REF! error value

Caller Property Example

This example displays information about how Visual Basic was called.

```
Select Case TypeName(Application.Caller)
   Case "Range"
      v = Application.Caller.Address
   Case "String"
      v = Application.Caller
   Case "Error"
      v = "Error"
   Case Else
      v = "unknown"
End Select
MsgBox "caller = " & v
```

Caption Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCaptionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCaptionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCaptionA "}

Returns text with one of several different meanings, depending on the object type to which it's applied. Read/write **String**, except as noted in the following table.

Object type	Meaning
Application	The name that appears in the title bar of the main Microsoft Excel window. If you don't set a name, or if you set the name to **Empty**, this property returns "Microsoft Excel." Read-only on the Macintosh.
AxisTitle	The axis title text.
Characters	The text of this range of characters.
ChartTitle	The chart title text.
DataLabel	The data label text.
Window	The name that appears in the title bar of the document window. When you set the name, you can use that name as the index to the **Windows** property (see the second example).

Caption Property Example

This example sets the name that appears in the title bar of the main Microsoft Excel window to be a custom name (this can be done only in Windows; on the Macintosh, the **Caption** property of the **Application** object is read-only).

```
Application.Caption = "Blue Sky Airlines Reservation System"
```

This example sets the name of the first window in the active workbook to be "Consolidated Balance Sheet." This name is then used as the index to the **Windows** property.

```
ActiveWorkbook.Windows(1).Caption = "Consolidated Balance Sheet"
ActiveWorkbook.Windows("Consolidated Balance Sheet") _
    .ActiveSheet.Calculate
```

Charts Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChartsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChartsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChartsA "}

Application object: Returns a **Sheets** collection that represents all the chart sheets in the active workbook. Read-only.

Workbook object: Returns a **Sheets** collection that represents all the chart sheets in the specified workbook. Read-only.

Using this property without an object qualifier returns all chart sheets in the active workbook.

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

Charts Property Example

This example sets the text for the title of Chart1.

```
With Charts("Chart1")
    .HasTitle = True
    .ChartTitle.Text = "First Quarter Sales"
End With
```

This example deletes every chart sheet in the active workbook.

```
ActiveWorkbook.Charts.Delete
```

This example hides Chart1, Chart3, and Chart5.

```
Charts(Array("Chart1", "Chart3", "Chart5")).Visible = False
```

ClipboardFormats Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproClipboardFormatsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproClipboardFormatsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproClipboardFormatsA "}

Returns the formats that are currently on the Clipboard, as an array of numeric values. To determine whether a particular format is on the Clipboard, compare each element in the array with the appropriate constant listed in the Remarks section.

Syntax

expression.**ClipboardFormats(*Index*)**

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

Index Optional **Variant**. The array element to be returned. If this argument is omitted, the property returns the entire array of formats that are currently on the Clipboard. For more information, see the Remarks section.

Remarks

This property is available both in Windows and on the Macintosh. Some formats may be available only in Windows or only on the Macintosh.

This property returns an array of numeric values. To determine whether a particular format is on the Clipboard compare each element of the array with one of the following **XlClipboardFormat** constants:

xlClipboardFormatBIFF
xlClipboardFormatBIFF2
xlClipboardFormatBIFF3
xlClipboardFormatBIFF4
xlClipboardFormatBinary
xlClipboardFormatBitmap
xlClipboardFormatCGM
xlClipboardFormatCSV
xlClipboardFormatDIF
xlClipboardFormatDspText
xlClipboardFormatEmbeddedObject
xlClipboardFormatEmbedSource
xlClipboardFormatLink
xlClipboardFormatLinkSource
xlClipboardFormatLinkSourceDesc
xlClipboardFormatMovie
xlClipboardFormatNative
xlClipboardFormatObjectDesc
xlClipboardFormatObjectLink
xlClipboardFormatOwnerLink
xlClipboardFormatPICT
xlClipboardFormatPrintPICT
xlClipboardFormatRTF
xlClipboardFormatScreenPICT
xlClipboardFormatStandardFont
xlClipboardFormatStandardScale
xlClipboardFormatSYLK
xlClipboardFormatTable
xlClipboardFormatText
xlClipboardFormatToolFace
xlClipboardFormatToolFacePICT
xlClipboardFormatVALU
xlClipboardFormatWK1

ClipboardFormats Property Example

This example displays a message box if the Clipboard contains a rich-text format (RTF) object. You can create an RTF object by copying text from a Word document.

```
aFmts = Application.ClipboardFormats
For Each fmt In aFmts
   If fmt = xlClipboardFormatRTF Then
      MsgBox "Clipboard contains rich text"
   End If
Next
```

Colors Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColorsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColorsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColorsA "}

Returns or sets colors in the palette for the workbook. The palette has 56 entries, each represented by an RGB value. Read/write **Variant**.

Syntax

expression.**Colors(*Index*)**

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

Index Optional **Variant**. The color number (from 1 to 56). If this argument isn't specified, this method returns an array that contains all 56 of the colors in the palette.

Colors Property Example

This example sets the color palette for the active workbook to be the same as the palette for Book2.xls.

```
ActiveWorkbook.Colors = Workbooks("BOOK2.XLS").Colors
```

This example sets color five in the color palette for the active workbook.

```
ActiveWorkbook.Colors(5) = RGB(255, 0, 0)
```

Comment Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCommentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCommentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCommentA "}

Range object: Returns a **Comment** object that represents the comment associated with the cell in the upper-left corner of the range. Read-only.

Scenario object: Returns or sets the comment associated with the scenario. The comment text cannot exceed 255 characters. Read/write **String**.

Comment Property Example

This example sets the comment for scenario one on Sheet1.

```
Worksheets("Sheet1").Scenarios(1).Comment = _
    "Worst case July 1993 sales"
```

Comments Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCommentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCommentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCommentsA "}

Returns a **Comments** collection that represents all the comments for the specified worksheet. Read-only.

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

Comments Property Example

This example deletes all comments added by Jean Selva on the active sheet.

```
For Each c in ActiveSheet.Comments
   If c.Author = "Jean Selva" Then c.Delete
Next
```

DisplayAlerts Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayAlertsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayAlertsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayAlertsA "}

True if Microsoft Excel displays certain alerts and messages while a macro is running. Read/write **Boolean**.

Remarks

The default value is **True**. Set this property to **False** if you don't want to be disturbed by prompts and alert messages while a macro is running; any time a message requires a response, Microsoft Excel chooses the default response.

If you set this property to **False**, Microsoft Excel doesn't automatically set it back to **True** when your macro stops running. Your macro should always set the property back to **True** when it stops running.

Note This behavior is different from previous versions of Microsoft Excel. In earlier versions, the **DisplayAlerts** property was automatically reset to **True** when the macro stopped running. If you have old code that relies on this behavior, you should change your code to explicitly set the property back to **True** at the end of the macro.

DisplayAlerts Property Example

This example closes the workbook Book1.xls and doesn't prompt the user to save changes. Any changes to Book1.xls aren't saved.

```
Application.DisplayAlerts = False
Workbooks("BOOK1.XLS").Close
Application.DisplayAlerts = True
```

This example suppresses the message that otherwise appears when you initiate a DDE channel to an application that's not running.

```
Application.DisplayAlerts = False
channelNumber = Application.DDEInitiate( _
   app:="WinWord", _
   topic:="C:\WINWORD\FORMLETR.DOC")
Application.DisplayAlerts = True
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
Application.DisplayAlerts = True
```

DisplayNoteIndicator Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayNoteIndicatorC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayNoteIndicatorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayNoteIndicatorA "}

True if cells containing notes display cell tips and contain note indicators (small dots in their upper-right corners). Read/write **Boolean**.

DisplayNoteIndicator Property Example

This example hides note indicators.

```
Application.DisplayNoteIndicator = False
```

End Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmproEndC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndA "}

Returns a **Range** object that represents the cell at the end of the region that contains the source range. Equivalent to pressing END+UP ARROW, END+DOWN ARROW, END+LEFT ARROW, or END+RIGHT ARROW. Read-only.

Syntax

expression.**End(*Direction*)**

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

Direction Required **Long**. The direction in which to move. Can be one of the following **XlDirection** constants: **xlToLeft**, **xlToRight**, **xlUp**, or **xlDown**.

End Property Example

This example selects the cell at the top of column B in the region that contains cell B4.

```
Range("B4").End(xlUp).Select
```

This example selects the cell at the end of row 4 in the region that contains cell B4.

```
Range("B4").End(xlToRight).Select
```

This example extends the selection from cell B4 to the last cell in row four that contains data.

```
Worksheets("Sheet1").Activate
Range("B4", Range("B4").End(xlToRight)).Select
```

FileConverters Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFileConvertersC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFileConvertersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFileConvertersA "}

Returns information about installed file converters. Returns **Null** if there are no converters installed. Read-only **Variant**.

Syntax

expression.**FileConverters(*Index1, Index2*)**

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

Index1 Optional **Variant**. The long name of the converter, including the file-type search string in Windows (for example, "Lotus 1-2-3 Files (*.wk*)").

Index2 Optional **Variant**. The path of the converter DLL or code resource.

Remarks

If you don't specify the index arguments, this property returns an array that containing information about all the installed file converters. Each row in the array contains information about a single file converter, as shown in the following table.

Column	Contents
1	The long name of the converter
2	The path of the converter DLL or code resource
3	The file-extension search string in Windows, or the four-character file type on the Macintosh

FileConverters Property Example

This example displays a message if the Multiplan file converter is installed.

```
installedCvts = Application.FileConverters
foundMultiplan = False
If Not IsNull(installedCvts) Then
   For arrayRow = 1 To UBound(installedCvts, 1)
      If installedCvts(arrayRow, 1) Like "*Multiplan*" Then
         foundMultiplan = True
         Exit For
      End If
   Next arrayRow
End If
If foundMultiplan = True Then
   MsgBox "Multiplan converter is installed"
Else
   MsgBox "Multiplan converter is not installed"
End If
```

Height Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHeightC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHeightA "}

Returns or sets the height of an object, in points. Read/write **Long**, except as shown in the following table.

Remarks

The meaning of the **Height** property depends on the specified object.

Object type	Height
Application	The height of the main application window. On the Macintosh, this is always equal to the total height of the screen, in points. Setting this value to something else on the Macintosh will have no effect. In Windows, if the window is minimized, this property is read-only and refers to the height of the icon. If the window is maximized, this property cannot be set. Use the **WindowState** property to determine the window state.
Axis, **LegendEntry**, **LegendKey**	The height of the object. Read-only.
Range	The height of the range. Read-only.
Window	The height of the window. Use the **UsableHeight** property to determine the maximum size for the window. You cannot set this property if the window is maximized or minimized. Use the **WindowState** property to determine the window state.
ChartArea, **ChartObject**, **Legend**, **OLEObject**, **PlotArea**, **Shape**, **ShapeRange**	The height of the object.

Height Property Example

This example sets the height of the emdbedded chart.

```
Worksheets("Sheet1").ChartObjects(1).Height = 288
```

International Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInternationalC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInternationalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInternationalA "}

Returns information about the current country and international settings. Read-only **Variant**.

Syntax

expression.**International(*Index*)**

Elements

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

Index Required **Long**. The setting to be returned. Can be one of the following **XlApplicationInternational** constants.

Index	Type	Meaning
xlCountryCode	**Long**	Country version of Microsoft Excel.
xlCountrySetting	**Long**	Current country setting in the Windows Control Panel, or the country number as determined by your Macintosh system software.
xlDecimalSeparator	**String**	Decimal separator.
xlThousandsSeparator	**String**	Zero or thousands separator.
xlListSeparator	**String**	List separator.
xlUpperCaseRowLetter	**String**	Uppercase row letter (for R1C1-style references).
xlUpperCaseColumnLetter	**String**	Uppercase column letter.
xlLowerCaseRowLetter	**String**	Lowercase row letter.
xlLowerCaseColumnLetter	**String**	Lowercase column letter.
xlLeftBracket	**String**	Character used instead of the left bracket ([) in R1C1-style relative references.
xlRightBracket	**String**	Character used instead of the right bracket (]) in R1C1-style references.
xlLeftBrace	**String**	Character used instead of the left brace ({) in array literals.
xlRightBrace	**String**	Character used instead of the right brace (}) in array literals.
xlColumnSeparator	**String**	Character used to separate columns in array literals.
xlRowSeparator	**String**	Character used to separate rows in array literals.
xlAlternateArraySeparator	**String**	Alternate array item separator to use if the current array separator is the same as the decimal separator.
xlDateSeparator	**String**	Date separator (*/* in U.S. version).
xlTimeSeparator	**String**	Time separator (: in U.S. version).
xlYearCode	**String**	Year symbol in number formats (y in U.S. version).
xlMonthCode	**String**	Month symbol (m in U.S. version).
xlDayCode	**String**	Day symbol (d in U.S. version).

xlHourCode	**String**	Hour symbol (h in U.S. version).
xlMinuteCode	**String**	Minute symbol (m in U.S. version).
xlSecondCode	**String**	Second symbol (s in U.S. version).
xlCurrencyCode	**String**	Currency symbol ($ in U.S. version).
xlGeneralFormatName	**String**	Name of the General number format.
xlCurrencyDigits	**Long**	Number of decimal digits to use in currency formats.
xlCurrencyNegative	**Long**	Currency format for negative currency values: 0 = ($x) or (x$) 1 = -$x or -x$ 2 = $-x or x-$ 3 = $x- or x$- Note that the position of the currency symbol is determined by **xlCurrencyBefore**.
xlNoncurrencyDigits	**Long**	Number of decimal digits to use in noncurrency formats.
xlMonthNameChars	**Long**	Always returns three for backwards compatibility. In Microsoft Excel 97, short month names are read from Microsoft Windows and can have any length.
xlWeekdayNameChars	**Long**	Always returns three for backwards compatibility. In Microsoft Excel 97, short weekday names are read from Microsoft Windows and can have any length.
xlDateOrder	**Long**	Order of date elements: 0 = month-day-year 1 = day-month-year 2 = year-month-day
xl24HourClock	**Boolean**	**True** if using 24-hour time, **False** if using 12-hour time.
xlNonEnglishFunctions	**Boolean**	**True** if not displaying functions in English.
xlMetric	**Boolean**	**True** if using the metric system, **False** if using the English measurement system.
xlCurrencySpaceBefore	**Boolean**	**True** if a space is added before the currency symbol.
xlCurrencyBefore	**Boolean**	**True** if the currency symbol precedes the currency values, **False** if it follows them.
xlCurrencyMinusSign	**Boolean**	**True** if using a minus sign for negative numbers, **False** if using parentheses.
xlCurrencyTrailingZeros	**Boolean**	**True** if trailing zeros are displayed for zero currency values.
xlCurrencyLeadingZeros	**Boolean**	**True** if leading zeros are displayed for zero currency values.
xlMonthLeadingZero	**Boolean**	**True** if a leading zero is displayed in months (when months are displayed as numbers).
xlDayLeadingZero	**Boolean**	**True** if a leading zero is displayed in days.
xl4DigitYears	**Boolean**	**True** if using four-digit years, **False** if using two-digit years.
xlMDY	**Boolean**	**True** if the date order is month-day-year for

		dates displayed in the long form, **False** if the date order is day-month-year.
xlTimeLeadingZero	**Boolean**	**True** if a leading zero is displayed in times.

International Property Example

This example displays the international decimal separator.

```
MsgBox "The decimal separator is " & _
   Application.International(xlDecimalSeparator)
```

Left Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLeftC "}
{ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLeftX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLeftA "}

Returns or sets the position of the specified object, in points. Read/write **Long**, except as shown in the following table.

Remarks

The meaning of the **Left** property depends on the specified object.

Object	Meaning
Application	The distance from the left edge of the screen to the left edge of the main Microsoft Excel window.
Axis, **LegendEntry**, **LegendKey**	The distance from the left edge of the object to the left edge of the chart area. Read-only.
Range	The distance from the left edge of column A to the left edge of the range. If the range is discontinuous, the first area is used. If the range is more than one column wide, the leftmost column in the range is used. Read-only.
Window	The distance from the left edge of the client area to the left edge of the window
AxisTitle, **ChartArea**, **ChartTitle**, **DataLabel**, **Legend**, **OLEObject**, **PlotArea**, **Shape**, **ShapeRange**	The distance from the left edge of the object to the left edge of column A (on a worksheet) or the left edge of the chart area (on a chart).

If the window is maximized, `Application.Left` returns a negative number that varies based on the width of the window border. Setting `Application.Left` to 0 (zero) will make the window a tiny bit smaller than it would be if the application window were maximized. In other words, if `Application.Left` is 0 (zero), the left border of the main Microsoft Excel window will just barely be visible on the screen.

On the Macintosh, `Application.Left` is always 0 (zero). Setting this value to something else on the Macintosh will have no effect.

In Windows, if the Microsoft Excel window is minimized, `Application.Left` controls the position of the window icon.

Left Property Example

This example aligns the left edge of the embedded chart with the left edge of column B.

```
With Worksheets("Sheet1")
   .ChartObjects(1).Left = .Columns("B").Left
End With
```

PrintQuality Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPrintQualityC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPrintQualityX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPrintQualityA "}

Returns or sets the print quality.

Syntax

expression.**PrintQuality(*Index*)**

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

Index Optional **Variant**. Horizontal print quality (1) or vertical print quality (2). Some printers may not support vertical print quality. If you don't specify this argument, the **PrintQuality** method returns (or can be set to) a two-element array that contains both horizontal and vertical print quality.

PrintQuality Method Example

This example sets print quality on a printer with nonsquare pixels. The array specifies both horizontal and vertical print quality. This example may cause an error, depending on the printer driver you're using.

```
Worksheets("Sheet1").PageSetup.PrintQuality = Array(240, 140)
```

This example displays the current setting for horizontal print quality.

```
MsgBox "Horizontal Print Quality is " & _
   Worksheets("Sheet1").PageSetup.PrintQuality(1)
```

Recipients Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRecipientsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRecipientsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRecipientsA "}

Returns or sets the recipients on the routing slip.

Syntax

expression.**Recipients(*Index*)**

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

Index Optional **Variant**. The recipient. If this argument isn't specified, the method returns (or can be set to) an array that contains all recipients.

Remarks

The order of the recipient list defines the delivery order if the routing delivery option is **xlOneAfterAnother**.

If a routing slip is in progress, only those recipients who haven't already received and routed the document are returned or set.

Recipients Method Example

This example sends the open workbook to three recipients, one after the other.

```
With ThisWorkbook
    .HasRoutingSlip = True
    With .RoutingSlip
        .Delivery = xlOneAfterAnother
        .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
        .Subject = "Here is the workbook"
        .Message = "Here is the workbook. What do you think?"
        .ReturnWhenDone = True
End With
    .Route
End With
```

RegisteredFunctions Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRegisteredFunctionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRegisteredFunctionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRegisteredFunctionsA "}

Returns information about functions in either dynamic-link libraries (DLLs) or code resources that were registered with the REGISTER or REGISTER.ID macro functions. Read-only **Variant**.

Syntax

expression.**RegisteredFunctions(*Index1, Index2*)**

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

Index1 Optional **Variant**. The name of the DLL or code resource.

Index2 Optional **Variant**. The name of the function.

Remarks

If you don't specify the index arguments, this property returns an array that contains a list of all registered functions. Each row in the array contains information about a single function, as shown in the following table.

Column	Contents
1	The name of the DLL or code resource
2	The name of the procedure in the DLL or code resource
3	Strings specifying the data types of the return values, and the number and data types of the arguments

If there are no registered functions, this property returns **Null**.

RegisteredFunctions Property Example

This example creates a list of registered functions, placing one registered function in each row on Sheet1. Column A contains the full path and file name of the DLL or code resource, column B contains the function name, and column C contains the argument data type code.

```
theArray = Application.RegisteredFunctions
If IsNull(theArray) Then
   MsgBox "No registered functions"
Else
   For i = LBound(theArray) To UBound(theArray)
      For j = 1 To 3
         Worksheets("Sheet1").Cells(i, j).Formula = theArray(i, j)
      Next j
   Next i
End If
```

ReplacementList Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthReplacementListC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthReplacementListX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthReplacementListA":1}

Returns the array of AutoCorrect replacements.

Syntax

expression.**ReplacementList(*Index*)**

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

Index Optional **Variant**. The row of the array of AutoCorrect replacements to be returned. The row is returned as a one-dimensional array with two elements: The first element is the text in column 1, and the second element is the text in column 2. If ***Index*** is out of range, this method fails.

If ***Index*** is not specified, this method returns a two-dimensional array. Each row in the array contains one replacement, as shown in the following table.

Column	Contents
1	The text to be replaced
2	The replacement text

Remarks

Use the **AddReplacement** method to add an entry to the replacement list.

ReplacementList Method Example

This example searches the replacement list for "Temperature" and displays the replacement entry if it exists.

```
repl = Application.AutoCorrect.ReplacementList
For x = 1 To UBound(repl)
      If repl(x, 1) = "Temperature" Then MsgBox repl(x, 2)
Next
```

Subject Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSubjectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSubjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSubjectA "}

Returns or sets the subject for the mailer or routing slip. Read/write **String**.

Remarks

The subject for the **RoutingSlip** object is used as the subject for mail messages used to route the workbook.

On the Macintosh, PowerTalk requires that there be a subject before the mailer can be sent.

Subject Property Example

This example sets the subject for a routing slip for the open workbook. To run this example, you must have Microsoft Exchange or Microsoft Mail for the Macintosh installed.

```
With ThisWorkbook
    .HasRoutingSlip = True
    With .RoutingSlip
        .Delivery = xlOneAfterAnother
        .Recipients = Array("Adam Bendel", "Jean Selva", "Bernard Gabor")
        .Subject = "Here is the workbook"
        .Message = "Here is the workbook. What do you think?"
        .ReturnWhenDone = True
End With
    .Route
End With
```

This example sets the subject for a mailer in the PowerTalk mail system (Macintosh only). To run this example, you must have the PowerTalk mail system installed.

```
With Workbooks(1)
    .HasMailer = True
    With .Mailer
        .Subject = "Here is the workbook"
        .ToRecipients = Array("Jean")
        .CCRecipients = Array("Adam", "Bernard")
        .BCCRecipients = Array("Chris")
        .Enclosures = Array("TestFile")
    End With
    .SendMailer
End With
```

Top Property

Returns or sets the position of the specified object, in points. Read/write **Long**, except as shown in the following table.

Remarks

The meaning of the **Top** property depends on the specified object.

Object	Meaning
Application	The distance from the top edge of the screen to the top edge of the main Microsoft Excel window. In Windows, if the application window is minimized, this property controls the position of the window icon (anywhere on the screen). On the Macintosh, the value is always 0 (zero); setting the value to something else will have no effect.
Range	The distance from the top edge of row 1 to the top edge of the range. If the range is discontinuous, the first area is used. If the range is more than one row high, the top (lowest numbered) row in the range is used. Read-only.
Window	The distance from the top edge of the window to the top edge of the usable area (below the menus, any toolbars docked at the top, and the formula bar). You cannot set this property for a maximized window. Use the **WindowState** property to return or set the state of the window.
Axis, **AxisTitle**, **ChartArea**, **ChartObject**, **ChartTitle**, **DataLabel**, **Legend**, **LegendEntry**, **LegendKey**, **OLEObject**, **PlotArea**	The distance from the top edge of the object to the top of row 1 (on a worksheet) or the top of the chart area (on a chart). Read-only for **Axis**, **LegendEntry**, and **LegendKey**.

Top Property Example

This example aligns the top of the embedded chart with the top of row two.

```
With Worksheets("Sheet1")
    .ChartObjects(1).Top = .Rows(2).Top
End With
```

This example expands the active window to the maximum size available (assuming that the window isn't already maximized).

```
With ActiveWindow
    .WindowState = xlNormal
    .Top = 1
    .Left = 1
    .Height = Application.UsableHeight
    .Width = Application.UsableWidth
End With
```

Width Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWidthC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWidthA "}

Returns or sets an object's width, in points. Read/write **Long**, except as shown in the following table.

Remarks

The meaning of the **Width** property depends on the specified object.

Object	Description
Application	The distance from the left edge of the application window to the right edge of the application window.
Axis, **LegendEntry**, **LegendKey**	The width of the object. Read-only.
Range	The width of the range.
Window	The width of the window. Use the **UsableWidth** property to determine the maximum size for the window. You cannot set this property if the window is maximized or minimized. Use the **WindowState** property to determine the window state.
ChartArea, **ChartObject**, **Legend**, **OLEObject**, **PlotArea**, **Shape**, **ShapeRange**, **Window**	The width of the object.

On the Macintosh, `Application.Width` is always equal to the total width of the screen, in points. Setting this value to any other value has no effect.

In Windows, if the window is minimized, `Application.Width` is read-only and returns the width of the window icon.

Width Property Example

This example sets the width of the embedded chart.

```
Worksheets("Sheet1").ChartObjects(1).Width = 360
```

This example expands the active window to the maximum size available (assuming that the window isn't maximized).

```
With ActiveWindow
    .WindowState = xlNormal
    .Top = 1
    .Left = 1
    .Height = Application.UsableHeight
    .Width = Application.UsableWidth
End With
```

Worksheets Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWorksheetsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWorksheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWorksheetsA "}

Application object: Returns a **Sheets** collection that represents all the worksheets in the active workbook. Read-only.

Workbook object: Returns a **Sheets** collection that represents all the worksheets in the specified workbook. Read-only.

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

Remarks

Using this property without an object qualifier returns all the worksheets in the active workbook.

This property doesn't return macro sheets; use the **Excel4MacroSheets** property or the **Excel4IntlMacroSheets** property to return those sheets.

Worksheets Property Example

This example displays the value in cell A1 on Sheet1 in the active workbook.

```
MsgBox Worksheets("Sheet1").Range("A1").Value
```

This example displays the name of each worksheet in the active workbook.

```
For Each ws In Worksheets
   MsgBox ws.Name
Next ws
```

This example adds a new worksheet to the active workbook and then sets the name of the worksheet.

```
Set newSheet = Worksheets.Add
newSheet.Name = "current Budget"
```

Quit Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthQuitC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthQuitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthQuitA "}

Quits Microsoft Excel. Doesn't run any Auto_Close macros before quitting.

Syntax

expression.**Quit**

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

Remarks

If unsaved workbooks are open when you use this method, Microsoft Excel displays a dialog box asking whether you want to save the changes. You can prevent this by saving all workbooks before using the **Quit** method or by setting the **DisplayAlerts** property to **False**. When this property is **False**, Microsoft Excel doesn't display the dialog box when you quit with unsaved workbooks; it quits without saving them.

If you set the **Saved** property for a workbook to **True** without saving the workbook to the disk, Microsoft Excel will quit without asking you to save the workbook.

Quit Method Example

This example saves all open workbooks and then quits Microsoft Excel.

```
For Each w In Application.Workbooks
   w.Save
Next w
Application.Quit
```

Subtotals Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSubtotalsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSubtotalsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSubtotalsA "}

Returns or sets subtotals displayed with the specified field. Valid only for nondata fields.

Syntax

expression.**Subtotals(*Index*)**

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

Index Optional **Variant**. A subtotal index, as shown in the following table. If this argument is omitted, the **Subtotals** method returns an array that contains a Boolean value for each subtotal.

Index	Meaning
1	Automatic
2	Sum
3	Count
4	Average
5	Max
6	Min
7	Product
8	Count Nums
9	StdDev
10	StdDevp
11	Var
12	Varp

If an index is **True**, the field shows that subtotal. If index 1 (Automatic) is **True**, all other values are set to **False**.

Subtotals Method Example

This example sets the field that contains the active cell to show Sum subtotals.

```
Worksheets("Sheet1").Activate
ActiveCell.PivotField.Subtotals(2) = True
```

Protect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthProtectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthProtectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthProtectA "}

Protects a chart or worksheet (Syntax 1) or a workbook (Syntax 2) so that it cannot be modified.

Syntax 1

expression.**Protect(*Password, DrawingObjects, Contents, Scenarios, UserInterfaceOnly*)**

Syntax 2

expression.**Protect(*Password, Structure, Windows*)**

expression Required. An expression that returns a **Chart** or **Worksheet** object (Syntax 1) or a **Workbook** object (Syntax 2).

Password Optional **Variant**. A string that specifies a case-sensitive password for the sheet or workbook. If this argument is omitted, you can unprotect the sheet or workbook without using a password. Otherwise, you must specify the password to unprotect the sheet or workbook. If you forget the password, you cannot unprotect the sheet or workbook. It's a good idea to keep a list of your passwords and their corresponding document names in a safe place.

DrawingObjects Optional **Variant**. **True** to protect shapes. The default value is **False**.

Contents Optional **Variant**. **True** to protect contents. For a chart, this protects the entire chart. For a worksheet, this protects the individual cells. The default value is **True**.

Scenarios Optional **Variant**. **True** to protect scenarios. This argument is valid only for worksheets. The default value is **True**.

Structure Optional **Variant**. **True** to protect the structure of the workbook (the relative position of the sheets). The default value is **False**.

UserInterfaceOnly Optional **Variant**. **True** to protect the user interface, but not macros. If this argument is omitted, protection applies both to macros and to the user interface.

Windows Optional **Variant**. **True** to protect the workbook windows. If this argument is omitted, the windows aren't protected.

Remarks

If you apply the **Protect** method with the ***UserInterfaceOnly*** argument set to **True** to a worksheet and then save the workbook, the entire worksheet (not just the interface) will be fully protected when you reopen the workbook. To unprotect the worksheet but re-enable user interface protection after the workbook is opened, you must again apply the **Protect** method with ***UserInterfaceOnly*** set to **True**.

Protect Method Example

This example protects the active workbook.

```
ActiveWorkbook.Protect Password := "drowssap"
```

ProtectContents Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectContentsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectContentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectContentsA "}

True if the contents of the sheet are protected. For a chart, this protects the entire chart. For a worksheet, this protects the individual cells. Read-only **Boolean**.

ProtectContents Property Example

This example displays a message box if the contents of Sheet1 are protected.

```
If Worksheets("Sheet1").ProtectContents = True Then
   MsgBox "The contents of Sheet1 are protected."
End If
```

ProtectDrawingObjects Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProtectDrawingObjectsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProtectDrawingObjectsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProtectDrawingObjectsA "}

True if shapes are protected. Read-only **Boolean**.

ProtectDrawingObjects Property Example

This example displays a message box if the shapes on Sheet1 are protected.

```
If Worksheets("Sheet1").ProtectDrawingObjects = True Then
   MsgBox "The shapes on Sheet1 are protected."
End If
```

Text Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTextA "}

Returns or sets the text for the specified object. Read-only **String** for the **Range** object, read/write **String** for all other objects.

For information about using the **Text** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Text Property Example

This example sets the text for the chart title of Chart1.

```
With Charts("Chart1")
    .HasTitle = True
    .ChartTitle.Text = "First Quarter Sales"
End With
```

This example sets the axis title text for the category axis in Chart1.

```
With Charts("Chart1").Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Text = "Month"
End With
```

This example illustrates the difference between the **Text** and **Value** properties of cells that contain formatted numbers.

```
Set c = Worksheets("Sheet1").Range("B14")
c.Value = 1198.3
c.NumberFormat = "$#,##0_);($#,##0)"
MsgBox c.Value
MsgBox c.Text
```

Activate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthActivateC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthActivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthActivateA "}

Activates the object, as shown in the following table.

Object	Description
Chart, **ChartObject**	Makes this chart the active chart.
Worksheet	Makes this sheet the active sheet. Equivalent to clicking the sheet's tab.
OLEObject	Activates the object.
Pane	Activates the pane. If the pane isn't in the active window, the window that the pane belongs to will also be activated. You cannot activate a frozen pane.
Range	Activates a single cell, which must be inside the current selection. To select a range of cells, use the **Select** method.
Window	Brings the window to the front of the z-order. This won't run any Auto_Activate or Auto_Deactivate macros that might be attached to the workbook (use the **RunAutoMacros** method to run those macros).
Workbook	Activates the first window associated with the workbook. This won't run any Auto_Activate or Auto_Deactivate macros that might be attached to the workbook (use the **RunAutoMacros** method to run those macros).

Syntax

expression.**Activate**

expression Required. An expression that returns an object in the Applies To list.

Activate Method Example

This example activates Sheet1.

```
Worksheets("Sheet1").Activate
```

This example selects cells A1:C3 on Sheet1 and then makes cell B2 the active cell.

```
Worksheets("Sheet1").Activate
Range("A1:C3").Select
Range("B2").Activate
```

This example activates Book4.xls. If Book4.xls has multiple windows, the example activates the first window, Book4.xls:1.

```
Workbooks("BOOK4.XLS").Activate
```

Index Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIndexC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIndexA "}

Returns the index number of the object within the collection of similar objects. Read-only **Long**.

For information about using the **Index** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Index Property Example

This example displays the tab number of the sheet name that you type. For example, if Chart1 is the third tab in the active workbook, the example displays "3" in a message box.

```
sheetname = InputBox("Type a sheet name, such as Sheet12")
MsgBox "This sheet is tab number " & Sheets(sheetname).Index
```

Cells Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCellsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCellsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCellsA "}

Application object: Returns a **Range** object that represents all the cells on the active worksheet. If the active document isn't a worksheet, this property fails. Read-only.

Range object: Returns a **Range** object that represents the cells in the specified range (in other words, it does nothing). Read-only.

Worksheet object: Returns a **Range** object that represents all the cells on the worksheet (not just the cells that are currently in use). Read-only.

Remarks

Because the **Item** property is the default property for the **Range** object, you can specify the row and column index immediately after the **Cells** keyword. For more information, see the **Item** property and the examples for this topic.

Using this property without an object qualifier returns a **Range** object that represents all the cells on the active worksheet.

Cells Property Example

This example sets the font size for cell C5 on Sheet1 to 14 points.

```
Worksheets("Sheet1").Cells(5, 3).Font.Size = 14
```

This example clears the formula in cell one on Sheet1.

```
Worksheets("Sheet1").Cells(1).ClearContents
```

This example sets the font and font size for every cell on Sheet1 to 8-point Arial.

```
With Worksheets("Sheet1").Cells.Font
   .Name = "Arial"
   .Size = 8
End With
```

This example loops through cells A1:J4 on Sheet1. If a cell contains a value less than 0.001, the example replaces that value with 0 (zero).

```
For rwIndex = 1 to 4
   For colIndex = 1 to 10
      With Worksheets("Sheet1").Cells(rwIndex, colIndex)
         If .Value < .001 Then .Value = 0
      End With
   Next colIndex
Next rwIndex
```

This example sets the font style for cells A1:C5 on Sheet1 to italic.

```
Worksheets("Sheet1").Activate
Range(Cells(1, 1), Cells(5, 3)).Font.Italic = True
```

This example scans a column of data named "myRange." If a cell has the same value as the cell immediately above it, the example displays the address of the cell that contains the duplicate data.

```
Set r = Range("myRange")
For n = 1 To r.Rows.Count
   If r.Cells(n, 1) = r.Cells(n + 1, 1) Then
      MsgBox "Duplicate data in " & r.Cells(n + 1, 1).Address
   End If
Next n
```

Returning an Object from a Collection

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlhowReturningAnObjectC"}

The **Item** property returns a single object from a collection. The following example sets the `firstBook` variable to a **Workbook** object that represents workbook one.

```
Set firstBook = Workbooks.Item(1)
```

The **Item** property is the default property for most collections, so you can write the same statement more concisely by omitting the **Item** keyword.

```
Set firstBook = Workbooks(1)
```

For more information about a specific collection, see the Help topic for that collection or the **Item** property for the collection.

Named Objects

Although you can usually specify an integer value with the **Item** property, it may be more convenient to return an object by name. Before you can use a name with the **Item** property, you must name the object. Most often, this is done by setting the object's **Name** property. The following example creates a named worksheet in the active workbook and then refers to the worksheet by name.

```
ActiveWorkbook.Worksheets.Add.Name = "a new sheet"
With Worksheets("a new sheet")
   .Range("a5:a10").Formula = "=rand()"
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 an **XlBordersIndex** constant with the **Item** property of the **Borders** collection to return a single border.

The following example sets the bottom border of cells A1:G1 on Sheet1 to a double line.

```
Worksheets("Sheet1").Range("a1:g1"). _
    Borders.Item(xlEdgeBottom).LineStyle = xlDouble
```

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** property is the default property for most collections in Microsoft Excel, so the following two statements are identical:

```
ActiveWorkbook.Worksheets.Item(1).Cells.Item(1, 1).Value = 5
ActiveWorkbook.Worksheets(1).Cells(1, 1).Value = 5
```

Note Default properties and methods are indicated by an asterisk following their names in the **Object Browser**.

Add Method (FormatConditions Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddFormatConditionsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddFormatConditionsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddFormatConditionsObjA"}

Adds a new conditional format. Returns a **FormatCondition** object that represents the new conditional format.

Syntax

expression.**Add(*Type, Operator, Formula1, Formula2*)**

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

Type Required **Long**. Specifies whether the conditional format is based on a cell value or an expression. Can be either of the following **XlFormatConditionType** constants: **xlCellValue** or **xlExpression**.

Operator Optional **Variant**. The conditional format operator. Can be one of the following **XlFormatConditionOperator** constants: **xlBetween**, **xlEqual**, **xlGreater**, **xlGreaterEqual**, **xlLess**, **xlLessEqual**, **xlNotBetween**, or **xlNotEqual**. If ***Type*** is **xlExpression**, the ***Operator*** argument is ignored.

Formula1 Optional **Variant**. The value or expression associated with the conditional format. Can be a constant value, a string value, a cell reference, or a formula.

Formula2 Optional **Variant**. The value or expression associated with the second part of the conditional format when ***Operator*** is **xlBetween** or **xlNotBetween** (otherwise, this argument is ignored). Can be a constant value, a string value, a cell reference, or a formula.

Remarks

You cannot define more than three conditional formats for a range. Use the **Modify** method to modify an existing conditional format, or use the **Delete** method to delete an existing format before adding a new one.

Add Method (FormatConditions Collection) Example

This example adds a conditional format to cells E1:E10.

```
With Worksheets(1).Range("e1:e10").FormatConditions _
        .Add(xlCellValue, xlGreater, "=$a$1")
    With .Borders
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = 6
    End With
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

Modify Method (FormatCondition Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthModifyFormatConditionObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthModifyFormatConditionObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthModifyFormatConditionObjA"}

Modifies an existing conditional format.

Syntax

expression.**Modify**(***Type, Operator, Formula1, Formula2***)

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

Type Required **Long**. Specifies whether the conditional format is based on a cell value or an expression. Can be either of the following **XlFormatConditionType** constants: **xlCellValue** or **xlExpression**.

Operator Optional **Variant**. The conditional format operator. Can be one of the following **XlFormatConditionOperator** constants: **xlBetween**, **xlEqual**, **xlGreater**, **xlGreaterEqual**, **xlLess**, **xlLessEqual**, **xlNotBetween**, or **xlNotEqual**. If ***Type*** is **xlExpression**, the ***Operator*** argument is ignored.

Formula1 Optional **Variant**. The value or expression associated with the conditional format. Can be a constant value, a string value, a cell reference, or a formula.

Formula2 Optional **Variant**. The value or expression associated with the second part of the conditional format when ***Operator*** is **xlBetween** or **xlNotBetween**. Can be a constant value, a string value, a cell reference, or a formula.

Modify Method (FormatCondition Object) Example

This example modifies an existing conditional format for cells E1:E10.

```
Worksheets(1).Range("e1:e10").FormatConditions(1) _
    .Modify xlCellValue, xlLess, "=$a$1"
```

FormatConditions Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormatConditionsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormatConditionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormatConditionsA"}

Returns a **FormatConditions** collection that represents all the conditional formats for the specified range. Read-only.

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

FormatConditions Property Example

This example modifies an existing conditional format for cells E1:E10.

```
Worksheets(1).Range("e1:e10").FormatConditions(1) _
    .Modify xlCellValue, xlLess, "=$a$1"
```

Formula1 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormula1C"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormula1X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormula1A"}

Returns the value or expression associated with the conditional format or data validation. Can be a constant value, a string value, a cell reference, or a formula. Read-only **String**.

Formula1 Property Example

This example changes the formula for conditional format one for cells E1:E10 if the formula specifies "less than 5."

```
With Worksheets(1).Range("e1:e10").FormatConditions(1)
   If .Operator = xlLess And .Formula1 = "5" Then
      .Modify xlCellValue, xlLess, "10"
   End If
End With
```

Formula2 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormula2C"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormula2X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormula2A"}

Returns the value or expression associated with the second part of a conditional format or data validation. Used only when the data validation conditional format **Operator** property is **xlBetween** or **xlNotBetween**. Can be a constant value, a string value, a cell reference, or a formula. Read-only **String**.

Formula2 Property Example

This example changes the formula for conditional format one for cells E1:E10 if the formula specifies "between 5 and 10"

```
With Worksheets(1).Range("e1:e10").FormatConditions(1)
   If .Operator = xlBetween And .Formula1 = "5" And .Formula2 = "10" Then
      .Modify xlCellValue, xlLess, "10"
   End If
End With
```

Operator Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOperatorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOperatorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOperatorA"}

Returns the operator for the conditional format or data validation. Can be one of the following **XlFormatConditionOperator** constants: **xlBetween**, **xlEqual**, **xlGreater**, **xlGreaterEqual**, **xlLess**, **xlLessEqual**, **xlNotBetween**, or **xlNotEqual**. Read-only **Long**.

Operator Property Example

This example changes the formula for conditional format one for cells E1:E10 if the formula specifies "less than 5."

```
With Worksheets(1).Range("e1:e10").FormatConditions(1)
   If .Operator = xlLess And .Formula1 = "5" Then
      .Modify xlCellValue, xlBetween, "5", "15"
   End If
End With
```

GetChartElement Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGetChartElementC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGetChartElementX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGetChartElementA"}

Returns information about the chart element at specified X and Y coordinates. This method is unusual in that you specify values for only the first two arguments. Microsoft Excel fills in the other arguments, and your code should examine those values when the method returns.

Syntax

expression.**GetChartElement(*X, Y, ElementID, Arg1, Arg2*)**

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

X Required **Long**. The X coordinate of the chart element.

Y Required **Long**. The Y coordinate of the chart element.

ElementID Required **Long**. When the method returns, this argument contains the **XLChartItem** value of the chart element at the specified coordinates. For more information, see the "Remarks" section.

Arg1 Required **Long**. When the method returns, this argument contains information related to the chart element. For more information, see the "Remarks" section.

Arg2 Required **Long**. When the method returns, this argument contains information related to the chart element. For more information, see the "Remarks" section.

Remarks

The value of ***ElementID*** after the method returns determines whether ***Arg1*** and ***Arg2*** contain any information, as shown in the following table.

ElementID	Arg1	Arg2
xlChartArea	None	None
xlChartTitle	None	None
xlPlotArea	None	None
xlLegend	None	None
xlFloor	None	None
xlWalls	None	None
xlCorners	None	None
xlDataTable	None	None
xlSeries	SeriesIndex	PointIndex
xlDataLabel	SeriesIndex	PointIndex
xlTrendline	SeriesIndex	TrendLineIndex
xlErrorBars	SeriesIndex	None
xlXErrorBars	SeriesIndex	None
xlYErrorBars	SeriesIndex	None
xlLegendEntry	SeriesIndex	None
xlLegendKey	SeriesIndex	None
xlAxis	AxisIndex	AxisType
xlMajorGridlines	AxisIndex	AxisType
xlMinorGridlines	AxisIndex	AxisType
xlAxisTitle	AxisIndex	AxisType
xlUpBars	GroupIndex	None
xlDownBars	GroupIndex	None

xlSeriesLines	GroupIndex	None
xlHiLoLines	GroupIndex	None
xlDropLines	GroupIndex	None
xlRadarAxisLabels	GroupIndex	None
xlShape	ShapeIndex	None
xlNothing	None	None

The following table describes the meaning of ***Arg1*** and ***Arg2*** after the method returns.

Argument	Description
SeriesIndex	Specifies the offset within the **Series** collection for a specific series.
PointIndex	Specifies the offset within the **Points** collection for a specific point within a series. A value of – 1 indicates that all data points are selected.
TrendlineIndex	Specifies the offset within the **Trendlines** collection for a specific trendline within a series.
AxisIndex	Specifies whether the axis is primary (0) or secondary (1).
AxisType	Specifies the axis type: Category (0), Value (1), or Series (2).
GroupIndex	Specifies the offset within the **ChartGroups** collection for a specific chart group.
ShapeIndex	Specifies the offset within the **Shapes** collection for a specific shape.

GetChartElement Method Example

This example warns the user if she moves the mouse over the chart legend.

```
Private Sub Chart_MouseMove(ByVal Button As Long, _
        ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)
    Dim IDNum As Long
    Dim a As Long
    Dim b As Long

    ActiveChart.GetChartElement X, Y, IDNum, a, b
    If IDNum = xlLegendEntry Then _
        MsgBox "WARNING: Move away from the legend"
End Sub
```

HasBorderHorizontal Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasBorderHorizontalC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasBorderHorizontalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasBorderHorizontalA"}

True if the chart data table has horizontal cell borders. Read/write **Boolean**.

HasBorderHorizontal Property Example

This example causes the embedded chart data table to be displayed with an outline border and no cell borders.

```
With Worksheets(1).ChartObjects(1).Chart
    .HasDataTable = True
    With .DataTable
        .HasBorderHorizontal = False
        .HasBorderVertical = False
        .HasBorderOutline = True
    End With
End With
```

HasBorderOutline Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasBorderOutlineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasBorderOutlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasBorderOutlineA"}

True if the chart data table has outline borders. Read/write **Boolean**.

HasBorderOutline Property Example

This example causes the embedded chart data table to be displayed with an outline border and no cell borders.

```
With Worksheets(1).ChartObjects(1).Chart
    .HasDataTable = True
    With .DataTable
        .HasBorderHorizontal = False
        .HasBorderVertical = False
        .HasBorderOutline = True
    End With
End With
```

HasBorderVertical Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasBorderVerticalC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasBorderVerticalX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasBorderVerticalA"}

True if the chart data table has vertical cell borders. Read/write **Boolean**.

HasBorderVertical Property Example

This example causes the embedded chart data table to be displayed with an outline border and no cell borders.

```
With Worksheets(1).ChartObjects(1).Chart
    .HasDataTable = True
    With .DataTable
        .HasBorderHorizontal = False
        .HasBorderVertical = False
        .HasBorderOutline = True
    End With
End With
```

HasDataTable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasDataTableC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasDataTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasDataTableA"}

True if the chart has a data table. Read/write **Boolean**.

HasDataTable Property Example

This example causes the embedded chart data table to be displayed with an outline border and no cell borders.

```
With Worksheets(1).ChartObjects(1).Chart
    .HasDataTable = True
    With .DataTable
        .HasBorderHorizontal = False
        .HasBorderVertical = False
        .HasBorderOutline = True
    End With
End With
```

HasLeaderLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasLeaderLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasLeaderLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasLeaderLinesA"}

True if the series has leader lines. Read/write **Boolean**.

HasLeaderLines Property Example

This example adds data labels and blue leader lines to series one on the pie chart.

```
With Worksheets(1).ChartObjects(1).Chart.SeriesCollection(1)
    .HasDataLabels = True
    .DataLabels.Position = xlLabelPositionBestFit
    .HasLeaderLines = True
    .LeaderLines.Border.ColorIndex = 5
End With
```

LeaderLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLeaderLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLeaderLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLeaderLinesA"}

Returns a **LeaderLines** object that represents the leader lines for the series. Read-only.

LeaderLines Property Example

This example adds data labels and blue leader lines to series one on the pie chart.

```
With Worksheets(1).ChartObjects(1).Chart.SeriesCollection(1)
    .HasDataLabels = True
    .DataLabels.Position = xlLabelPositionBestFit
    .HasLeaderLines = True
    .LeaderLines.Border.ColorIndex = 5
End With
```

Location Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthLocationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthLocationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthLocationA"}

Moves the chart to a new location.

Syntax

expression.**Location(*Where, Name*)**

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

Where Required **Long**. Where to move the chart. Can be one of the following **XlChartLocation** constants: **xlLocationAsNewSheet**, **xlLocationAsObject**, or **xlLocationAutomatic**.

Name Optional **Variant**; required if ***Where*** is **xlLocationAsObject**. The name of the sheet where the chart will be embedded if ***Where*** is **xlLocationAsObject** or the name of the new sheet if ***Where*** is **xlLocationAsNewSheet.**

Location Method Example

This example moves the embedded chart to a new chart sheet named "Monthly Sales."

```
Worksheets(1).ChartObjects(1).Chart _
    .Location xlLocationAsNewSheet, "Monthly Sales"
```

MajorUnitScale Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMajorUnitScaleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMajorUnitScaleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMajorUnitScaleA"}

Returns or sets the major unit scale value for the category axis when the **CategoryType** property is set to **xlTimeScale**. Can be one of the following **XlTimeUnit** constants: **xlDays**, **xlMonths**, or **xlYears**. Read/write **Long**.

MajorUnitScale Property Example

This example sets the category axis to use a time scale and sets the major and minor units.

```
With Charts(1).Axes(xlCategory)
    .CategoryType = xlTimeScale
    .MajorUnit = 5
    .MajorUnitScale = xlDays
    .MinorUnit = 1
    .MinorUnitScale = xlDays
End With
```

MarkerSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarkerSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarkerSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarkerSizeA"}

Returns or sets the data-marker size, in points. Read/write **Long**.

MarkerSize Property Example

This example sets the data-marker size for all data markers on series one.

```
Worksheets(1).ChartObjects(1).Chart.SeriesCollection(1).MarkerSize = 10
```

MinorUnitScale Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinorUnitScaleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinorUnitScaleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinorUnitScaleA"}

Returns or sets the minor unit scale value for the category axis when the **CategoryType** property is set to **xlTimeScale**. Can be one of the following **XlTimeUnit** constants: **xlDays**, **xlMonths**, or **xlYears**. Read/write **Long**.

MinorUnitScale Property Example

This example sets the category axis to use a time scale and sets the major and minor units.

```
With Charts(1).Axes(xlCategory)
    .CategoryType = xlTimeScale
    .MajorUnit = 5
    .MajorUnitScale = xlDays
    .MinorUnit = 1
    .MinorUnitScale = xlDays
End With
```

NewSeries Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthNewSeriesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthNewSeriesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthNewSeriesA"}

Creates a new series. Returns a **Series** object that represents the new series.

Syntax

expression.**NewSeries()**

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

NewSeries Method Example

This example adds a new series to chart one.

```
Set ns = Charts(1).SeriesCollection.NewSeries
```

PersonalViewListSettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPersonalViewListSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPersonalViewListSettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPersonalViewListSettingsA"}

True if filter and sort settings for lists are included in the user's personal view of the shared workbook. Read/write **Boolean**.

PersonalViewListSettings Property Example

This example removes print settings and filter and sort settings from the user's personal view of workbook two.

```
With Workbooks(2)
    .PersonalViewListSettings = False
    .PersonalViewPrintSettings = False
End With
```

PersonalViewPrintSettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPersonalViewPrintSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPersonalViewPrintSettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPersonalViewPrintSettingsA"}

True if print settings are included in the user's personal view of the shared workbook. Read-write **Boolean**.

PersonalViewPrintSettings Property Example

This example removes print settings and filter and sort settings from the user's personal view of workbook two.

```
With Workbooks(2)
    .PersonalViewListSettings = False
    .PersonalViewPrintSettings = False
End With
```

PrintComments Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintCommentsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintCommentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintCommentsA"}

Returns or sets the way comments are printed with the sheet. Can be one of the following **XlPrintLocation** constants: **xlPrintInPlace**, **xlPrintNoComments**, or **xlPrintSheetEnd**. Read-write **Long**.

PrintComments Property Example

This example causes comments to be printed as end notes when worksheet one is printed.

```
Worksheets(1).PageSetup.PrintComments = xlPrintSheetEnd
```

RemoveUser Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRemoveUserC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRemoveUserX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRemoveUserA"}

Disconnects the specified user from the shared workbook.

Syntax

expression.**RemoveUser(*Index*)**

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

Index Required **Long**. The user index.

RemoveUser Method Example

This example disconnects user two from the shared workbook.

```
Workbooks(2).RemoveUser 2
```

ResetAllPageBreaks Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthResetAllPageBreaksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthResetAllPageBreaksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthResetAllPageBreaksA"}

Resets all page breaks on the specified worksheet.

Syntax

expression.**ResetAllPageBreaks()**

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

ResetAllPageBreaks Method Example

This example resets all page breaks on worksheet one.

```
Worksheets(1).ResetAllPageBreaks
```

SaveWorkspace Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSaveWorkspaceC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSaveWorkspaceX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSaveWorkspaceA"}

Saves the current workspace.

Syntax

expression.**SaveWorkspace(*Filename*)**

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

Filename Optional **Variant**. The saved file name.

SaveWorkspace Method Example

This example saves the current workspace as "saved workspace.xlw".

```
Application.SaveWorkspace "saved workspace"
```

ScrollArea Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproScrollAreaC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproScrollAreaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproScrollAreaA"}

Returns or sets the range where scrolling is allowed, as an A1-style range reference. Cells outside the scroll area cannot be selected. Read/write **String**.

Remarks

Set this property to the empty string ("") to enable cell selection for the entire sheet.

ScrollArea Property Example

This example sets the scroll area for worksheet one.

```
Worksheets(1).ScrollArea = "a1:f10"
```

ShowNegativeBubbles Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowNegativeBubblesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowNegativeBubblesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowNegativeBubblesA"}

True if negative bubbles are shown for the chart group. Valid only for bubble charts. Read/write **Boolean**.

ShowNegativeBubbles Property Example

This example makes negative bubbles visible for chart group one.

```
Worksheets(1).ChartObjects(1).Chart _
    .ChartGroups(1).ShowNegativeBubbles = True
```

UserControl Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUserControlC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUserControlX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUserControlA"}

True if the application was created or started by the user. **False** if the application was created or started programmatically using the **CreateObject** or **GetObject** functions. Read/write **Boolean**.

Remarks

When the **UserControl** property is **False** for an object, that object is released when the last programmatic reference to the object is released. If this property is **False**, Microsoft Excel quits when the last object in the session is released.

UserControl Property Example

This example displays the status of the **UserControl** property.

```
If ThisWorkbook.UserControl Then
   MsgBox "This workbook was created by the user"
Else
   MsgBox "This workbook was created programmatically"
End If
```

HPageBreaks Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHPageBreaksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHPageBreaksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHPageBreaksA"}

Returns an **HPageBreaks** collection that represents the horizontal page breaks on the sheet. Read-only.

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

HPageBreaks Property Example

This example displays the number of full-screen and print-area horizontal page breaks.

```
For Each pb in Worksheets(1).HPageBreaks
   If pb.Extent = xlPageBreakFull Then
      cFull = cFull + 1
   Else
      cPartial = cPartial + 1
   End If
Next
MsgBox cFull & " full-screen page breaks, " & cPartial & _
   " print-area page breaks"
```

Extent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproExtentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproExtentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproExtentA"}

Returns the type of the specified page break: full-screen or only within a print area. Can be either of the following **XlPageBreakExtent** constants: **xlPageBreakFull** or **xlPageBreakPartial**. Read-only **Long**.

Extent Property Example

This example displays the total number of full-screen and print-area horizontal page breaks.

```
For Each pb in Worksheets(1).HPageBreaks
   If pb.Extent = xlPageBreakFull Then
      cFull = cFull + 1
   Else
      cPartial = cPartial + 1
   End If
Next
MsgBox cFull & " full-screen page breaks, " & cPartial & _
   " print-area page breaks"
```

Location Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLocationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLocationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLocationA"}

Returns or sets the cell (a **Range** object) that defines the page-break location. Horizontal page breaks are aligned with the top edge of the location cell; vertical page breaks are aligned with the left edge of the location cell. Read/write **Range**.

Location Property Example

This example moves the horizontal page-break location.

```
Worksheets(1).HPageBreaks(1).Location = Worksheets(1).Range("e5")
```

VPageBreaks Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVPageBreaksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVPageBreaksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVPageBreaksA"}

Returns a **VPageBreaks** collection that represents the vertical page breaks on the sheet. Read-only.

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

VPageBreaks Property Example

This example displays the total number of full-screen and print-area vertical page breaks.

```
For Each pb in Worksheets(1).VPageBreaks
   If pb.Extent = xlPageBreakFull Then
      cFull = cFull + 1
   Else
      cPartial = cPartial + 1
   End If
Next
MsgBox cFull & " full-screen page breaks, " & cPartial & _
   " print-area page breaks"
```

MergeWorkbook Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthMergeWorkbookC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthMergeWorkbookX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthMergeWorkbookA"}

Merges changes from one workbook into an open workbook.

Syntax

expression.**MergeWorkbook(*Filename*)**

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

Filename Required **String**. The file name of the workbook that contains the changes to be merged into the open workbook.

MergeWorkbook Method Example

This example merges changes from Book1. xls into the active workbook.

```
ActiveWorkbook.MergeWorkbook "Book1.xls"
```

IndentLevel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIndentLevelC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIndentLevelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIndentLevelA"}

Returns or sets the indent level for the range or style. Can be an integer from 0 to 15. Read/write **Variant**.

Remarks

Using this property to set the indent level to a number less than 0 (zero) or greater than 15 causes an error.

IndentLevel Property Example

This example increases the indent level to 15 in cell A10.

```
With Range("a10")
    .IndentLevel = 15
End With
```

InsertIndent Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthInsertIndentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthInsertIndentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthInsertIndentA"}

Adds an indent to the specified range.

Syntax

expression.**InsertIndent(*InsertAmount*)**

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

InsertAmount Required **Long**. The amount to be added to the current indent.

Remarks

Using this method to set the indent level to a number less than 0 (zero) or greater than 15 causes an error.

Use the **IndentLevel** property to return the indent level for a range.

InsertIndent Method Example

This example decreases the indent level in cell A10.

```
With Range("a10")
    .InsertIndent -1
End With
```

MergeArea Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMergeAreaC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMergeAreaX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMergeAreaA"}

Returns a **Range** object that represents the merged range containing the specified cell. If the specified cell isn't in a merged range, this property returns the specified cell. Read-only **Variant**.

MergeArea Property Example

This example sets the value of the merged range that contains cell A3.

```
Set ma = Range("a3").MergeArea
If ma.Address = "$A$3" Then
   MsgBox "not merged"
Else
   ma.Cells(1, 1).Value = "42"
End If
```

MergeCells Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMergeCellsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMergeCellsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMergeCellsA"}

True if the range or style contains merged cells. Read/write **Variant**.

Remarks

When you select a range that contains merged cells, the resulting selection may be different from the intended selection. Use the **Address** property to check the address of the selected range.

MergeCells Property Example

This example sets the value of the merged range that contains cell A3.

```
Set ma = Range("a3").MergeArea
If Range("a3").MergeCells Then
   ma.Cells(1, 1).Value = "42"
End If
```

UnMerge Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUnMergeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUnMergeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUnMergeA"}

Separates a merged area into individual cells.

Syntax

expression.**UnMerge**

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

UnMerge Method Example

This example separates the merged range that contains cell A3.

```
With Range("a3")
   If .MergeCells Then
      .MergeArea.UnMerge
   Else
      MsgBox "not merged"
   End If
End With
```

AddIndent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAddIndentC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAddIndentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAddIndentA "}

This property is not used in U.S./English Microsoft Excel.

Merge Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthMergeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthMergeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthMergeA "}

Syntax 1: Merges the scenarios from another sheet into the **Scenarios** collection.

Syntax 2: Merges the styles from another workbook into the **Styles** collection.

Syntax 3: Creates a merged cell from the specified **Range** object.

Syntax 1

expression.**Merge(*Source*)**

Syntax 2

expression.**Merge(*Workbook*)**

Syntax 3

expression.**Merge(*Across*)**

expression Required. An expression that returns a **Scenarios** object (Syntax 1), **Styles** object (Syntax 2), or **Range** object (Syntax 3).

Source Required **Variant**. The name of the sheet that contains scenarios to be merged, or a **Worksheet** object that represents that sheet.

Workbook Required **Variant**. A **Workbook** object that represents the workbook containing styles to be merged.

Across Optional **Variant**. **True** to merge cells in each row in the specified range as separate merged cells. The default value is **False**.

Remarks

The value of a merged range is the value specified for the cell in the upper-left corner of the merged range.

Merge Method Example

This example merges the styles from the workbook Template.xls into the active workbook.

```
ActiveWorkbook.Styles.Merge Workbook:=Workbooks("TEMPLATE.XLS")
```

Borders Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBordersC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBordersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBordersA "}

Returns a **Borders** collection that represents the four borders of a style or a range of cells (including a range defined as part of a conditional format). Read-only.

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

Borders Property Example

This example sets the color of the bottom border of cell B2 on Sheet1 to red.

```
With Worksheets("Sheet1").Range("B2").Borders(xlBottom)
    .LineStyle = xlBorderLineStyleContinuous
    .Weight = xlThin
    .ColorIndex = 3
End With
```

Rotation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRotationC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRotationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRotationA "}

Chart object: Returns or sets the rotation of the 3-D chart view (the rotation of the plot area around the z-axis, in degrees). The value of this property must be from 0 to 360, except for 3-D bar charts, where the value must be from 0 to 44. The default value is 20. Applies only to 3-D charts. Read/write **Variant**.

Shape or **ShapeRange** object: Returns or sets the rotation of the shape, in degrees. Read/write **Single**.

Rotation Property Example

This example sets the rotation of Chart1 to 30 degrees. The example should be run on a 3-D chart.

```
Charts("Chart1").Rotation = 30
```

SizeRepresents Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSizeRepresentsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSizeRepresentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSizeRepresentsA"}

Returns or sets what the bubble size represents on a bubble chart. Can be either of the following **XlSizeRepresents** constants: **xlSizeIsArea** or **xlSizeIsWidth**. Read/write **Long**.

SizeRepresents Property Example

This example sets what the bubble size represents for chart group one.

```
Charts(1).ChartGroups(1).SizeRepresents = xlSizeIsWidth
```

ApplyPictToEnd Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproApplyPictToEndC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproApplyPictToEndX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproApplyPictToEndA"}

True if a picture is applied to the end of the point or all points in the series. Read/write **Boolean**.

ApplyPictToEnd Property Example

This example applies pictures to the end of all points in series one. The series must already have pictures applied to it (setting this property changes the picture orientation).

```
Charts(1).SeriesCollection(1).ApplyPictToEnd = True
```

ApplyPictToFront Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproApplyPictToFrontC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproApplyPictToFrontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproApplyPictToFrontA"}

True if a picture is applied to the front of the point or all points in the series. Read/write **Boolean**.

ApplyPictToFront Property Example

This example applies pictures to the front of all points in series one. The series must already have pictures applied to it (setting this property changes the picture orientation).

```
Charts(1).SeriesCollection(1).ApplyPictToFront = True
```

ApplyPictToSides Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproApplyPictToSidesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproApplyPictToSidesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproApplyPictToSidesA"}

True if a picture is applied to the sides of the point or all points in the series. Read/write **Boolean**.

ApplyPictToSides Property Example

This example applies pictures to the sides of all points in series one. The series must already have pictures applied to it (setting this property changes the picture orientation).

```
Charts(1).SeriesCollection(1).ApplyPictToSides = True
```

Assistant Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAssistantC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAssistantX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAssistantA"}

Returns an **Assistant** object for Microsoft Excel.

Remarks

Using this property without an object qualifier is equivalent to using `Application.Assistant`.

Assistant Property Example

This example makes the Office Assistant visible.

```
Assistant.Visible = True
```

AutoUpdateFrequency Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoUpdateFrequencyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoUpdateFrequencyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoUpdateFrequencyA"}

Returns or sets the number of minutes between automatic updates to the shared workbook. If this property is set to zero (0), updates occur only when the workbook is saved. Read/write **Long**.

AutoUpdateFrequency Property Example

This example causes the shared workbook to be automatically updated every three minutes.

```
ActiveWorkbook.AutoUpdateFrequency = 3
```

AutoUpdateSaveChanges Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoUpdateSaveChangesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoUpdateSaveChangesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoUpdateSaveChangesA"}

True if current changes to the shared workbook are posted to other users whenever the workbook is automatically updated. **False** if changes aren't posted (this workbook is still synchronized with changes made by other users). The default value is **True**. Read/write **Boolean**.

Remarks

The **AutoUpdateFrequency** property must be set to a value from 5 to 1440 for this property to take effect.

AutoUpdateSaveChanges Property Example

This example causes changes to the shared workbook to be posted to other users whenever the workbook is automatically updated.

```
ActiveWorkbook.AutoUpdateSaveChanges = True
```

CommandBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCommandBarsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCommandBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCommandBarsA"}

Returns a **CommandBars** object that represents the Microsoft Excel command bars. Read-only.

Remarks

Used with the **Application** object, this property returns the set of built-in and custom command bars available to the application.

When a workbook is embedded in another application and activated by the user by double-clicking the workbook, using this property with a **Workbook** object returns the set of Microsoft Excel command bars available within the other application. At all other times, using this property with a **Workbook** object returns **Nothing**.

There is no programmatic way to return the set of command bars attached to a workbook.

CommandBars Property Example

This example deletes all custom command bars that aren't visible.

```
For Each bar In Application.CommandBars
    If Not bar.BuiltIn And Not bar.Visible Then bar.Delete
Next
```

ConflictResolution Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproConflictResolutionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConflictResolutionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConflictResolutionA"}

Returns or sets the way conflicts are to be resolved whenever a shared workbook is updated. Read/write **Long**.

Can be one of the following **XlSaveConflictResolution** constants.

Constant	Description
xlLocalSessionChanges	The local user's changes are always accepted.
xlOtherSessionChanges	The local user's changes are always rejected.
xlUserResolution	A dialog box asks the user to resolve the conflict.

ConflictResolution Property Example

This example causes the local user's changes to be accepted whenever there's a conflict in the shared workbook.

```
ActiveWorkbook.ConflictResolution = xlLocalSessionChanges
```

DataTable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDataTableC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDataTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDataTableA"}

Returns a **DataTable** object that represents the chart data table. Read-only.

DataTable Property Example

This example adds a data table with an outline border to the embedded chart.

```
With Worksheets(1).ChartObjects(1).Chart
    .HasDataTable = True
    .DataTable.HasBorderOutline = True
End With
```

EnableEvents Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableEventsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableEventsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableEventsA"}

True if events are enabled for the specified object. Read/write **Boolean**.

EnableEvents Property Example

This example disables events before a file is saved so that the **BeforeSave** event doesn't occur.

```
Application.EnableEvents = False
ActiveWorkbook.Save
Application.EnableEvents = True
```

EnableResize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableResizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableResizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableResizeA"}

True if the window can be resized. Read/write **Boolean**.

EnableResize Property Example

This example sets the active window so that it cannot be resized.

```
ActiveWindow.EnableResize = False
```

EnableSelection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableSelectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableSelectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableSelectionA"}

Returns or sets what can be selected on the sheet. Can be one of the following **XlEnableSelection** constants: **xlNoRestrictions**, **xlNoSelection**, or **xlUnlockedCells**. Read/write **Long**.

Remarks

This property takes effect only when the worksheet is protected: **xlNoSelection** prevents any selection on the sheet, **xlUnlockedCells** allows only those cells whose **Locked** property is **False** to be selected, and **xlNoRestrictions** allows any cell to be selected.

EnableSelection Property Example

This example sets worksheet one so that nothing on it can be selected.

```
With Worksheets(1)
    .EnableSelection = xlNoSelection
    .Protect Contents:=True, UserInterfaceOnly:=True
End With
```

MergeLabels Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMergeLabelsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMergeLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMergeLabelsA"}

True if PivotTable outer-row item, column item, subtotal, and grand total labels use merged cells. Read-write **Boolean**.

MergeLabels Property Example

This example causes the PivotTable to use merged-cell outer-row item, column item, subtotal, and grand total labels.

```
Worksheets(1).PivotTables(1).MergeLabels = True
```

AutoShow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoShowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoShowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoShowA"}

Displays the number of top or bottom items for a PivotTable row, page, or column field.

Syntax

expression.**AutoShow(*Type, Range, Count, Field*)**

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

Type Required **Long**. Use **xlAutomatic** to cause the PivotTable to show the items that match the specified criteria. Use **xlManual** to disable this feature.

Range Required **Long**. The location at which to start showing items. Can be either of the following constants: **xlTop** or **xlBottom**.

Count Required **Long**. The number of items to be shown.

Field Required **String**. The name of the base data field.

AutoShow Method Example

This example shows only the top two companies, based on the sum of sales:

```
ActiveSheet.PivotTables("Pivot1").PivotFields("Company") _
    .AutoShow  xlAutomatic, xlTop, 2, "Sum of Sales"
```

ServerBased Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproServerBasedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproServerBasedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproServerBasedA"}

True if the PivotTable's data source is external and only the items matching the page field selection are retrieved. Read/write **Boolean**.

When this property is **True**, only records in the database that match the selected page field item are retrieved. From then on, whenever the user changes the page field selection, the newly selected page field item is passed to the query as a parameter, and the cache is refreshed.

This property cannot be set if any of the following conditions are true:

- The field is grouped.
- The data source isn't external.
- The cache is shared by two or more PivotTables.
- The field is a data type that cannot be server based (a memo field or an OLE object).

ServerBased Property Example

This example lists all the server-based page fields.

```
For Each fld in ActiveSheet.PivotTables(1).PageFields
   If fld.ServerBased = True Then
      r = r + 1
      Worksheets(2).Cells(r, 1).Value = fld.Name
   End If
Next
```

PivotFormulas Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotFormulasC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotFormulasX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotFormulasA"}

Returns a **PivotFormulas** object that represents the collection of pivot formulas for the PivotTable. Read-only.

Syntax

expression.**PivotFormulas()**

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

PivotFormulas Method Example

This example creates a list of pivot formulas for PivotTable one.

```
For Each pf in ActiveSheet.PivotTables(1).PivotFormulas
   r = r + 1
   Cells(r, 1).Value = pf.Formula
Next
```

AutoSort Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAutoSortC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutoSortX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutoSortA"}

Establishes automatic PivotTable field-sorting rules.

Syntax

expression.**AutoSort(*Order, Field*)**

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

Order Required **Long**. The sort order. Can be either of the following **XlSortOrder** constants: **xlAscending** or **xlDescending**. Can also be **xlManual** to disable automatic sorting.

Field Required **String**. The name of the sort key field.

AutoSort Method Example

This example sorts the Company field in descending order, based on the sum of sales.

```
ActiveSheet.PivotTables(1).PivotField("Company") _
    .AutoSort xlDescending, "Sum of Sales"
```

Selection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSelectionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSelectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSelectionA "}

Application object: Returns the selected object in the active window.

Window object: Returns the selected object in the specified window.

Remarks

The returned object type depends on the current selection (for example, if a cell is selected, this property returns a **Range** object). The **Selection** property returns **Nothing** if nothing is selected.

Using this property with no object qualifier is equivalent to using `Application.Selection`.

Selection Property Example

This example clears the selection on Sheet1 (assuming that the selection is a range of cells).

```
Worksheets("Sheet1").Activate
Selection.Clear
```

This example displays the Visual Basic object type of the selection.

```
Worksheets("Sheet1").Activate
MsgBox "The selection object type is " & TypeName(Selection)
```

PageBreak Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageBreakC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageBreakX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageBreakA "}

Returns or sets the location of a page break. Can be one of the following **XlPageBreak** constants: **xlPageBreakAutomatic**, **xlPageBreakManual**, or **xlPageBreakNone**. Read/write **Long**.

Remarks

This property can return the location of either automatic or manual page breaks, but it can only set the location of manual breaks (it can only be set to **xlPageBreakManual** or **xlPageBreakNone**).

To remove all manual page breaks on a worksheet, set `Cells.PageBreak` to **xlPageBreakNone**.

PageBreak Property Example

This example sets a manual page break above row 25 on Sheet1.

```
Worksheets("Sheet1").Rows(25).PageBreak = xlPageBreakManual
```

This example sets a manual page break to the left of column J on Sheet1.

```
Worksheets("Sheet1").Columns("J").PageBreak = xlPageBreakManual
```

This example deletes the two page breaks that were set in the preceding examples.

```
Worksheets("Sheet1").Rows(25).PageBreak = xlPageBreakNone
Worksheets("Sheet1").Columns("J").PageBreak = xlNone
```

RecordMacro Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRecordMacroC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRecordMacroX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRecordMacroA "}

Records code if the macro recorder is on.

Syntax

expression.**RecordMacro**(***BasicCode, XlmCode***)

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

BasicCode Optional **Variant**. A string that specifies the Visual Basic code that will be recorded if the macro recorder is recording into a Visual Basic module. The string will be recorded on one line. If the string contains a carriage return (ASCII character 10, or `Chr$(10)` in code), it will be recorded on more than one line.

XlmCode Optional **Variant**. This argument is ignored.

Remarks

The **RecordMacro** method cannot record into the active module (the module in which the **RecordMacro** method exists).

If ***BasicCode*** is omitted and the application is recording into Visual Basic, Microsoft Excel will record a suitable `Application.Run` statement.

To prevent recording (for example, if the user cancels your dialog box), call this function with two empty strings.

RecordMacro Method Example

This example records Visual Basic code.

```
Application.RecordMacro BasicCode:="Application.Run ""MySub"" "
```

CheckSpelling Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCheckSpellingC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCheckSpellingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCheckSpellingA "}

Syntax 1: Checks the spelling of an object. This form has no return value; Microsoft Excel displays the **Spelling** dialog box.

Syntax 2: Checks the spelling of a single word. Returns **True** if the word is found in one of the dictionaries, **False** if it isn't.

Syntax 1

expression.**CheckSpelling(*CustomDictionary, IgnoreUppercase, AlwaysSuggest, IgnoreInitialAlefHamza, IgnoreFinalYaa, SpellScript*)**

Syntax 2

expression.**CheckSpelling(*Word, CustomDictionary, IgnoreUppercase*)**

expression Required. An expression that returns an object in the Applies To list. Use the **Application** object to check a single word (Syntax 2).

CustomDictionary Optional **Variant**. A string that indicates the file name of the custom dictionary to examine if the word isn't found in the main dictionary. If this argument is omitted, the currently specified dictionary is used.

IgnoreUppercase Optional **Variant**. **True** to have Microsoft Excel ignore words that are all uppercase. **False** to have Microsoft Excel check words that are all uppercase. If this argument is omitted, the current setting will be used.

AlwaysSuggest Optional **Variant**. **True** to have Microsoft Excel display a list of suggested alternate spellings when an incorrect spelling is found. **False** to have Microsoft Excel wait for you to input the correct spelling. If this argument is omitted, the current setting will be used.

Word Required **String** (used with **Application** object only). The word you want to check.

IgnoreInitialAlefHamza Opional **Variant**. Not used in U.S./English Microsoft Excel.

IgnoreFinalYaa Opional **Variant**. Not used in U.S./English Microsoft Excel.

SpellScript Opional **Variant**. Not used in U.S./English Microsoft Excel.

Remarks

To check headers, footers, and objects on a worksheet, use this method on a **Worksheet** object.

To check only cells and notes, use this method with the object returned by the **Cells** method.

CheckSpelling Method Example

This example checks the spelling on Sheet1.

```
Worksheets("Sheet1").CheckSpelling
```

Ungroup Method (Range Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUngroupRangeObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUngroupRangeObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUngroupRangeObjA "}

Promotes a range in an outline (that is, decreases its outline level). The specified range must be a row or column, or a range of rows or columns. If the range is in a PivotTable, this method ungroups the items contained in the range.

Syntax

expression.**Ungroup**

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

Remarks

If the active cell is in a field header of a parent field, all the groups in that field are ungrouped and the field is removed from the PivotTable. When the last group in a parent field is ungrouped, the entire field is removed from the PivotTable.

Ungroup Method (Range Object) Example

This example ungroups the ORDER_DATE field.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
Set groupRange = pvtTable.PivotFields("ORDER_DATE").DataRange
groupRange.Cells(1).Ungroup
```

Position Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPositionC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPositionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPositionA "}

Returns or sets the position of the specified object, as shown in the following table. Read/write **Long**.

Object	Position
DataLabel, **DataLabels**	Position of the data label. Can be one of the following **XlDataLabelPosition** constants: **xlLabelPositionAbove**, **xlLabelPositionBelow**, **xlLabelPositionBestFit**, **xlLabelPositionCenter**, **xlLabelPositionCustom**, **xlLabelPositionInsideBase**, **xlLabelPositionInsideEnd**, **xlLabelPositionLeft**, **xlLabelPositionMixed**, **xlLabelPositionOutsideEnd**, or **xlLabelPositionRight**.
Legend	Position of the legend on the chart. Can be one of the following **XlLegendPosition** constants: **xlLegendPositionBottom**, **xlLegendPositionCorner**, **xlLegendPositionLeft**, **xlLegendPositionRight**, or **xlLegendPositionTop**.
PivotField	Position of the field (first, second, third, and so on) among all the fields in its orientation (Rows, Columns, Pages, Data).
PivotItem	Position of the item in its field, if the item is currently showing.

Position Property Example

This example moves the chart legend to the bottom of the chart.

```
Charts(1).Legend.Position = xlLegendPositionBottom
```

This example displays the position number of the pivot item that contains the active cell.

```
Worksheets("Sheet1").Activate
MsgBox "The active item is in position number " & _
   ActiveCell.PivotItem.Position
```

Orientation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOrientationC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOrientationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOrientationA "}

Returns or sets the object's orientation, as shown in the following table. Read-write **Long**.

Object	Orientation
ChartObject, **ChartObjects**	Embedded chart placement. Can be one of the following **XlPlacement** constants: **xlFreeFloating**, **xlMove**, or **xlMoveAndSize**.
PageSetup	Portrait or landscape printing mode. Can be either of the following **XlPageOrientation** constants: **xlLandscape** or **xlPortrait**.
PivotField	The location of the field in the PivotTable. Can be one of the following **XlPivotFieldOrientation** constants: **xlColumnField**, **xlDataField**, **xlHidden**, **xlPageField**, or **xlRowField**.
AxisTitle, **ChartTitle**, **DataLabel**, **Range**, **Style**, **TextFrame**	The text orientation. Can be an integer value from –90 to 90 degrees or one of the following **XlOrientation** constants: **xlDownward**, **xlHorizontal**, **xlUpward**, or **xlVertical**.
TickLabels	The text orientation. Can be an integer value from –90 to 90 degrees or one of the following **XlTickLabelOrientation** constants: **xlTickLabelOrientationAutomatic**, **xlTickLabelOrientationDownward**, **xlTickLabelOrientationHorizontal**, **xlTickLabelOrientationUpward**, or **xlTickLabelOrientationVertical**.

Orientation Property Example

This example displays the orientation for the ORDER_DATE field.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
Set pvtField = pvtTable.PivotFields("ORDER_DATE")
Select Case pvtField.Orientation
   Case xlHidden
      MsgBox "Hidden field"
   Case xlRowField
      MsgBox "Row field"
   Case xlColumnField
      MsgBox "Column field"
   Case xlPageField
      MsgBox "Page field"
   Case xlDataField
      MsgBox "Data field"
End Select
```

This example sets Sheet1 to be printed in landscape orientation.

```
Worksheets("Sheet1").PageSetup.Orientation = xlLandscape
```

BuiltinDocumentProperties Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBuiltinDocumentPropertiesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBuiltinDocumentPropertiesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBuiltinDocumentPropertiesA "}

Returns a **DocumentProperties** collection that represents all the built-in document properties for the specified workbook. Read-only.

Remarks

This property returns the entire collection of built-in document properties. Use the **Item** method to return a single member of the collection (a **DocumentProperty** object) by specifying either the name of the property or the collection index (as a number).

You can refer to document properties either by index value or by name. The following list shows the available built-in document property names:

Title	Creation Date	Company
Subject	Last Save Time	Number of Bytes
Author	Total Editing Time	Number of Lines
Keywords	Number of Pages	Number of Paragraphs
Comments	Number of Words	Number of Slides
Template	Number of Characters	Number of Notes
Last Author	Security	Number of Hidden Slides
Revision Number	Category	Number of Multimedia Clips
Application Name	Format	Hyperlink Base
Last Print Date	Manager	Number of Characters (with spaces)

Container applications aren't required to define values for every built-in document property. If Microsoft Excel doesn't define a value for one of the built-in document properties, reading the **Value** property for that document property causes an error.

Because the **Item** method is the default method for the **DocumentProperties** collection, the following statements are identical:

```
BuiltinDocumentProperties.Item(1)
BuiltinDocumentProperties(1)
```

Use the **CustomDocumentProperties** property to return the collection of custom document properties.

BuiltinDocumentProperties Property Example

This example displays the names of the built-in document properties as a list on worksheet one.

```
rw = 1
Worksheets(1).Activate
For Each p In ActiveWorkbook.BuiltinDocumentProperties
    Cells(rw, 1).Value = p.Name
    rw = rw + 1
Next
```

CustomDocumentProperties Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCustomDocumentPropertiesC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCustomDocumentPropertiesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCustomDocumentPropertiesA "}

Returns a **DocumentProperties** collection that represents all the custom document properties for the specified workbook. Read-only.

Remarks

This property returns the entire collection of custom document properties. Use the **Item** method to return a single member of the collection (a **DocumentProperty** object) by specifying either the name of the property or the collection index (as a number).

Because the **Item** method is the default method for the **DocumentProperties** collection, the following statements are identical:

```
CustomDocumentProperties.Item("Complete")
CustomDocumentProperties("Complete")
```

Use the **BuiltinDocumentProperties** property to return the collection of built-in document properties.

CustomDocumentProperties Property Example

This example displays the names and values of the custom document properties as a list on worksheet one.

```
rw = 1
Worksheets(1).Activate
For Each p In ActiveWorkbook.CustomDocumentProperties
    Cells(rw, 1).Value = p.Name
    Cells(rw, 2).Value = p.Value
    rw = rw + 1
Next
```

Post Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPostC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPostX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPostA "}

Posts the specified workbook to a public folder. This method works only with a Microsoft Exchange client connected to a Microsoft Exchange server.

Syntax

expression.**Post(*DestName*)**

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

DestName Optional **Variant**. This argument is ignored. The **Post** method prompts the user to specify the destination for the workbook.

Post Method Example

This example posts the active workbook.

```
ActiveWorkbook.Post
```

SaveAs Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSaveAsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSaveAsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSaveAsA "}

Saves changes to the sheet (Syntax 1) or workbook (Syntax 2) in a different file.

Syntax 1

expression.**SaveAs(*Filename, FileFormat, Password, WriteResPassword, ReadOnlyRecommended, CreateBackup, AddToMru, TextCodePage, TextVisualLayout*)**

Syntax 2

expression.**SaveAs(*Filename, FileFormat, Password, WriteResPassword, ReadOnlyRecommended, CreateBackup, AccessMode, ConflictResolution, AddToMru, TextCodePage, TextVisualLayout*)**

expression Required. An expression that returns a **Chart** or **Worksheet** object (Syntax 1) or a **Workbook** object (Syntax 2).

Filename Optional **Variant**. A string that indicates the name of the file to be saved. You can include a full path; if you don't, Microsoft Excel saves the file in the current folder.

FileFormat Optional **Variant**. The file format to use when you save the file. For a list of valid choices, see the **FileFormat** property.

Password Optional **Variant**. A case-sensitive string (no more than 15 characters) that indicates the protection password to be given to the file.

WriteResPassword Optional **Variant**. A string that indicates the write-reservation password for this file. If a file is saved with the password and the password isn't supplied when the file is opened, the file is opened as read-only.

ReadOnlyRecommended Optional **Variant**. **True** to display a message when the file is opened, recommending that the file be opened as read-only.

CreateBackup Optional **Variant**. **True** to create a backup file.

AccessMode Optional **Variant**. The workbook access mode. Can be one of the following **XlSaveAsAccessMode** constants: **xlShared** (shared list), **xlExclusive** (exclusive mode), or **xlNoChange** (don't change the access mode). If this argument is omitted, the access mode isn't changed. This argument is ignored if you save a shared list without changing the file name. To change the access mode, use the **ExclusiveAccess** method.

ConflictResolution Optional **Variant**. Specifies the way change conflicts are resolved if the workbook is a shared list. Can be one of the following **XlSaveConflictResolution** constants: **xlUserResolution** (display the conflict-resolution dialog box), **xlLocalSessionChanges** (automatically accept the local user's changes), or **xlOtherSessionChanges** (accept other changes instead of the local user's changes). If this argument is omitted, the conflict-resolution dialog box is displayed.

AddToMru Optional **Variant**. **True** to add this workbook to the list of recently used files. The default value is **False**.

TextCodePage Optional **Variant**. Not used in U.S. English Microsoft Excel.

TextVisualLayout Optional **Variant**. Not used in U.S. English Microsoft Excel.

SaveAs Method Example

This example creates a new workbook, prompts the user for a file name, and then saves the workbook.

```
Set NewBook = Workbooks.Add
Do
   fName = Application.GetSaveAsFilename
Loop Until fName <> False
NewBook.SaveAs Filename:=fName
```

Open Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOpenC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOpenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOpenA "}

Opens a workbook.

Syntax

expression.**Open(*FileName, UpdateLinks, ReadOnly, Format, Password, WriteResPassword, IgnoreReadOnlyRecommended, Origin, Delimiter, Editable, Notify, Converter, AddToMRU*)**

expression Required. An expression that returns a **Workbooks** or **RecentFile** object.

FileName Required **String**. The file name of the workbook to be opened.

UpdateLinks Optional **Variant**. Specifies the way links in the file are updated. If this argument is omitted, the user is prompted to specify how links will be updated. Otherwise, this argument is one of the values listed in the following table.

Value	Meaning
0	Doesn't update any references
1	Updates external references but not remote references
2	Updates remote references but not external references
3	Updates both remote and external references

If Microsoft Excel is opening a file in the WKS, WK1, or WK3 format and the ***UpdateLinks*** argument is 2, Microsoft Excel generates charts from the graphs attached to the file. If the argument is 0, no charts are created.

ReadOnly Optional **Variant**. True to open the workbook in read-only mode.

Format Optional **Variant**. If Microsoft Excel is opening a text file, this argument specifies the delimiter character, as shown in the following table. If this argument is omitted, the current delimiter is used.

Value	Delimiter
1	Tabs
2	Commas
3	Spaces
4	Semicolons
5	Nothing
6	Custom character (see the ***Delimiter*** argument)

Password Optional **Variant**. A string that contains the password required to open a protected workbook. If this argument is omitted and the workbook requires a password, the user is prompted for the password.

WriteResPassword Optional **Variant**. A string that contains the password required to write to a write-reserved workbook. If this argument is omitted and the workbook requires a password, the user will be prompted for the password.

IgnoreReadOnlyRecommended Optional **Variant**. **True** to have Microsoft Excel not display the read-only recommended message (if the workbook was saved with the **Read-Only Recommended** option).

Origin Optional **Variant**. If the file is a text file, this argument indicates where it originated (so that code pages and Carriage Return/Line Feed (CR/LF) can be mapped correctly). Can be one of the following **XlPlatform** constants: **xlMacintosh**, **xlWindows**, or **xlMSDOS**. If this argument is omitted, the current operating system is used.

Delimiter Optional **Variant**. If the file is a text file and the **Format** argument is 6, this argument is a string that specifies the character to be used as the delimiter. For example, use `Chr(9)` for tabs,

use "," for commas, use ";" for semicolons, or use a custom character. Only the first character of the string is used.

Editable Optional **Variant**. If the file is a Microsoft Excel 4.0 add-in, this argument is **True** to open the add-in so that it's a visible window. If this argument is **False** or omitted, the add-in is opened as hidden, and it cannot be unhidden. This option doesn't apply to add-ins created in Microsoft Excel 5.0 or later. If the file isn't an add-in, **True** prevents the running of any Auto_Open macros.

Notify Optional **Variant**. If the file cannot be opened in read/write mode, this argument is **True** to add the file to the file notification list. Microsoft Excel will open the file as read-only, poll the file notification list, and then notify the user when the file becomes available. If this argument is **False** or omitted, no notification is requested, and any attempts to open an unavailable file will fail.

Converter Optional **Variant**. The index of the first file converter to try when opening the file. The specified file converter is tried first; if this converter doesn't recognize the file, all other converters are tried. The converter index consists of the row numbers of the converters returned by the **FileConverters** property.

AddToMru Optional **Variant**. **True** to add this workbook to the list of recently used files. The default value is **False**.

Remarks

If the workbook being opened has any Auto_Open macros in it, they won't be run when you open the file from Visual Basic. If you want to run the Auto_Open macro, you must use the **RunAutoMacros** method.

Open Method Example

This example opens the workbook Analysis.xls and then runs its Auto_Open macro.

```
Workbooks.Open "ANALYSIS.XLS"
ActiveWorkbook.RunAutoMacros xlAutoOpen
```

MacroOptions Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthMacroOptionsC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthMacroOptionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthMacroOptionsA "}

Corresponds to options in the **Macro Options** dialog box.

Syntax

expression.**MacroOptions(*Macro, Description, HasMenu, MenuText, HasShortcutKey, ShortcutKey, Category, StatusBar, HelpContextID, HelpFile*)**

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

Macro Optional **Variant**. The macro name.

Description Optional **Variant**. The macro description.

HasMenu Optional **Variant**. This argument is ignored.

MenuText Optional **Variant**. This argument is ignored.

HasShortcutKey Optional **Variant**. **True** to assign a shortcut key to the macro (***ShortcutKey*** must also be specified). If this argument is **False**, no shortcut key is assigned to the macro. If the macro already has a shortcut key, setting this argument to **False** removes the shortcut key. The default value is **False**.

ShortcutKey Optional **Variant**. Required if ***HasShortcutKey*** is **True**; ignored otherwise. The shortcut key.

Category Optional **Variant**. An integer that specifies the macro function category (Financial, Date & Time, or User Defined, for example).

StatusBar Optional **Variant**. The status bar text for the macro.

HelpContextId Optional **Variant**. An integer that specifies the context ID for the Help topic assigned to the macro.

HelpFile Optional **Variant**. The name of the Help file that contains the Help topic defined by ***HelpContextId***.

MacroOptions Method Example

This example adds a shortcut key for the DoRand macro.

```
Application.MacroOptions Macro:="DoRand", _
   HasShortcutKey:=True, ShortcutKey:="Z"
```

NoteText Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthNoteTextC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthNoteTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthNoteTextA "}

Returns or sets the cell note associated with the cell in the upper-left corner of the range. Read/write **String**.

Cell notes have been replaced by range comments. For more information, see the **Comment** object.

Syntax

expression.**NoteText(*Text, Start, Length*)**

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

Text Optional **Variant**. The text to add to the note (up to 255 characters). The text is inserted starting at position ***Start***, replacing ***Length*** characters of the existing note. If this argument is omitted, this method returns the current text of the note starting at position ***Start***, for ***Length*** characters.

Start Optional **Variant**. The starting position for the text that's set or returned. If this argument is omitted, this method starts at the first character. To append text to the note, specify a number larger than the number of characters in the existing note.

Length Optional **Variant**. The number of characters to be set or returned. If this argument is omitted, Microsoft Excel sets or returns characters from the starting position to the end of the note (up to 255 characters). If there are more than 255 characters from ***Start*** to the end of the note, this method returns only 255 characters.

Remarks

To add a note that contains more than 255 characters, use this method once to specify the first 255 characters, and then use it again to append the remainder of the note (no more than 255 characters at a time).

NoteText Method Example

This example sets the cell note text for cell A1 on Sheet1.

```
Worksheets("Sheet1").Range("A1").NoteText "This may change!"
```

PictureType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPictureTypeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPictureTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPictureTypeA "}

Returns or sets the way pictures are displayed on a column or bar picture chart or on the walls and faces of a 3-D chart. Read/write **Variant**.

Can be one of the following **XlPictureType** constants.

Value	Meaning
xlStretch	Stretch the picture to reach the necessary value.
xlStack	Stack the pictures to reach the necessary value.
xlScale	Stack the pictures, but use the **PictureUnit** property to determine what unit each picture represents.
xlTile	Tile the pictures.

PictureType Property Example

This example sets series one in Chart1 to stretch pictures. The example should be run on a 2-D column chart with picture data markers.

```
Charts("Chart1").SeriesCollection(1).PictureType = xlStretch
```

PictureUnit Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPictureUnitC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPictureUnitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPictureUnitA "}

Returns or sets the unit for each picture on the chart if the **PictureType** property is set to **xlScale** (if not, this property is ignored). Read/write **Long**.

PictureUnit Property Example

This example sets series one in Chart1 to stack pictures and uses each picture to represent five units. The example should be run on a 2-D column chart with picture data markers.

```
With Charts("Chart1").SeriesCollection(1)
    .PictureType = xlScale
    .PictureUnit = 5
End With
```

Paste Method

Object	Description
Chart	Pastes chart data from the Clipboard into the specified chart.
Point or **Series**	Pastes a picture from the Clipboard as the marker on the selected point or series. **Paste** can be used on column, bar, line, or radar charts; it sets the **MarkerStyle** property to **xlPicture**.
SeriesCollection	Pastes data from the Clipboard into the specified series collection.
Worksheet	Pastes the contents of the Clipboard onto the sheet.
Walls or **Floor**	Pastes a picture from the Clipboard onto the walls or floor.

Group Method (Range Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGroupRangeObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGroupRangeObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGroupRangeObjA "}

When the **Range** object is in an outline, the **Group** method (Syntax 1) demotes the range (in other words, increases its outline level). The range should be an entire row or column, or a range of rows or columns.

When the **Range** object is a discontinuous range in a PivotTable, the **Group** method (Syntax 1) groups the range.

When the **Range** object represents a single cell in the pivot field's data range, the **Group** method (Syntax 2) performs numeric or date-based grouping in a pivot field.

Syntax 1

expression.**Group**

Syntax 2

expression.**Group(*Start, End, By, Periods*)**

expression Required. An expression that returns a **Range** object. For Syntax 2, the **Range** object must be a single cell in the pivot field's data range. If you attempt to apply this method to more than one cell, it will fail (without displaying an error message). To see how to use the **Group** method in this way, see the second example in the accompanying example topic.

Start Optional **Variant**. The first value to be grouped. If this argument is omitted or **True**, the first value in the field is used.

End Optional **Variant**. The last value to be grouped. If this argument is omitted or **True**, the last value in the field is used.

By Optional **Variant**. If the field is numeric, this argument specifies the size of each group.

If the field is a date, this argument specifies the number of days in each group if element 4 in the ***Periods*** array is **True** and all the other elements are **False**. Otherwise, this argument is ignored.

If this argument is omitted, Microsoft Excel automatically chooses a default group size.

Periods Optional **Variant**. An array of Boolean values that specify the period for the group, as shown in the following table.

Array element	Period
1	Seconds

2	Minutes
3	Hours
4	Days
5	Months
6	Quarters
7	Years

If an element in the array is **True**, a group is created for the corresponding time; if the element is **False**, no group is created. If the field isn't a date field, this argument is ignored.

Group Method (Range Object) Example

This example groups the field named "ORDER_DATE" by 10-day periods.

```
Set pvtTable = Worksheets("Sheet1").Range("A3").PivotTable
Set groupRange = pvtTable.PivotFields("ORDER_DATE").DataRange
groupRange.Cells(1).Group by:=10, periods:=Array(False, False, False, True,
False, False, False)
```

Pattern Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPatternC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPatternX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPatternA "}

Interior object: Returns or sets the interior pattern. Read/write **Long**.

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

xlPatternAutomatic
xlPatternChecker
xlPatternCrissCross
xlPatternDown
xlPatternGray16
xlPatternGray25
xlPatternGray50
xlPatternGray75
xlPatternGray8
xlPatternGrid
xlPatternHorizontal
xlPatternLightDown
xlPatternLightHorizontal
xlPatternLightUp
xlPatternLightVertical
xlPatternNone
xlPatternSemiGray75
xlPatternSolid
xlPatternUp
xlPatternVertical

FillFormat object: Returns or sets the fill pattern. 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
msoPatternDivits
msoPatternDottedGrid
msoPatternHorizontalBrick
msoPatternLargeCheckerBoard
msoPatternLargeConfetti
msoPatternLargeGrid
msoPatternLightDiamonds
msoPatternLightDownwardDiagonal
msoPatternLightHorizontal
msoPatternLightUpwardDiagonal
msoPatternLightVertical
msoPatternMixed
msoPatternNarrowHorizontal
msoPatternNarrowVertical
msoPatternOutlinedDiamonds
msoPatternPlaid
msoPatternShingles
msoPatternSmallCheckerBoard
msoPatternSmallConfetti
msoPatternSmallGrid
msoPatternSolidDiamonds
msoPatternSpheres
msoPatternTrellis
msoPatternWaves
msoPatternWavyLines
msoPatternWeave
msoPatternWideDownwardDiagonal
msoPatternWideUpwardDiagonal

Pattern Property Example

This example adds a crisscross pattern to the interior of cell A1 on Sheet1.

```
Worksheets("Sheet1").Range("A1").Interior.Pattern = xlPatternCrissCross
```

Value Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproValueC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproValueX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproValueA "}

The meaning of the **Value** property depends on the object to which it is applied, as shown in the following table.

Object	Value
Application	Always returns "Microsoft Excel". Read-only.
Borders	Synonym for `Borders.LineStyle`.
Name	A string containing the formula that the name is defined to refer to. The string is in A1-style notation in the language of the macro, and it begins with an equal sign. Read-only.
Parameter	The parameter value. For more information, see the **Parameter** object.
PivotField	The name of the specified field in the PivotTable.
PivotItem	The name of the specified item in the PivotTable field.
PivotTable	The name of the PivotTable.
Range	The value of the specified cell. If the cell is empty, **Value** returns the value **Empty** (use the **IsEmpty** function to test for this case). If the **Range** object contains more than one cell, returns an array of values (use the **IsArray** function to test for this case).
Style	The name of the specified style.
Validation	**True** if all the validation criteria are met (that is, if the range contains valid data).

Value Property Example

This example sets the value of cell A1 on Sheet1 to 3.14159.

```
Worksheets("Sheet1").Range("A1").Value = 3.14159
```

This example loops on cells A1:D10 on Sheet1. If one of the cells has a value less than 0.001, the code replaces the value with 0 (zero).

```
For Each c in Worksheets("Sheet1").Range("A1:D10")
   If c.Value < .001 Then
      c.Value = 0
   End If
Next c
```

Show Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthShowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthShowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthShowA "}

CustomView object (Syntax 1): Displays the custom view.

Range object (Syntax 1): Scrolls through the contents of the active window to move the range into view. The range must consist of a single cell in the active document.

Scenario object (Syntax 1): Shows the scenario by inserting its values on the worksheet. The affected cells are the changing cells of the scenario.

Dialog object (Syntax 2): Displays the built-in dialog box and waits for the user to input data.

Syntax 1

expression.**Show**

Syntax 2

object.**Show(*arg1, arg2, ..., arg30*)**

expression Required. For Syntax 1, an expression that returns an object in the Applies To list.

arg1, arg2, ..., arg30 Optional **Variant**. For built-in dialog boxes only, the initial arguments for the command. For more information, see the "Remarks" section.

Remarks

For built in dialog boxes, this method returns **True** if the user clicks OK, or it returns **False** if the user clicks Cancel.

You can use a single dialog box to change many properties at the same time. For example, you can use the **Format Cells** dialog box to change all the properties of the **Font** object.

For some built-in dialog boxes (the **Open** dialog box, for example), you can set initial values using ***arg1, arg2, ..., arg30***. To find the arguments to set, locate the corresponding dialog box constant in Built-In Dialog Box Argument Lists. For example, search for the **xlDialogOpen** constant to find the arguments for the **Open** dialog box. For more information about built-in dialog boxes, see the **Dialogs** collection.

Show Method Example

This example displays the **Open** dialog box.

```
Application.Dialogs(xlDialogOpen).Show
```

LineStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLineStyleC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLineStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLineStyleA "}

Returns or sets the line style for the border. Can be one of the following **XlLineStyle** constants: **xlContinuous**, **xlDash**, **xlDashDot**, **xlDashDotDot**, **xlDot**, **xlDouble**, **xlSlantDashDot**, or **xlLineStyleNone**. Read/write **Variant**.

LineStyle Property Example

This example puts a border around the chart area and the plot area of Chart1.

```
With Charts("Chart1")
    .ChartArea.Border.LineStyle = xlDasDot
    With .PlotArea.Border
        .LineStyle = xlDashDotDot
        .Weight = xlThick
    End With
End With
```

Shadow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShadowC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShadowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShadowA "}

True if the font is a shadow font or if the object has a shadow. Read/write **Boolean**.

Remarks

For the **Font** object, this property has no effect in Microsoft Windows, but its value is retained (it can be set and returned).

Shadow Property Example

This example adds a shadow to the title of Chart1.

```
Charts("Chart1").ChartTitle.Shadow = True
```

Range Property (Application, Range, or Worksheet Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRangeWorksheetObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRangeWorksheetObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRangeWorksheetObjA "}

Returns a **Range** object that represents a cell or a range of cells.

Syntax 1

expression.**Range(*Cell1*)**

Syntax 2

expression.**Range(*Cell1, Cell2*)**

expression Optional for **Application**, required for **Range** and **Worksheet**. An expression that returns an object in the Applies To list.

Cell1 Syntax 1: Required **Variant**. The name of the range. This must be an A1-style reference in the language of the macro. It can include the range operator (a colon), the intersection operator (a space), or the union operator (a comma). It can also include dollar signs, but they're ignored. You can use a local defined name in any part of the range. If you use a name, the name is assumed to be in the language of the macro.

Syntax 2: Required **Variant**. The cell in the upper-left corner of the range. Can be a **Range** object that contains a single cell, an entire column, or entire row), or it can be a string that names a single cell in the language of the macro.

Cell1, ***Cell2*** Optional **Variant**. The cell in the upper-left and lower-right corner of the range. Can be a **Range** object that contains a single cell, an entire column, or entire row, or it can be a string that names a single cell in the language of the macro.

Remarks

When used without an object qualifier, this property is a shortcut for `ActiveSheet.Range` (it returns a range from the active sheet; if the active sheet isn't a worksheet, the property fails).

When applied to a **Range** object, the property is relative to the **Range** object. For example, if the selection is cell C3, then `Selection.Range("B1")` returns cell D3 because it's relative to the **Range** object returned by the **Selection** property. On the other hand, the code `ActiveSheet.Range("B1")` always returns cell B1.

Range Property (Application, Range, or Worksheet Object) Example

This example sets the value of cell A1 on Sheet1 to 3.14159.

```
Worksheets("Sheet1").Range("A1").Value = 3.14159
```

This example creates a formula in cell A1 on Sheet1.

```
Worksheets("Sheet1").Range("A1").Formula = "=10*RAND()"
```

This example loops on cells A1:D10 on Sheet1. If one of the cells has a value less than 0.001, the code replaces that value with 0 (zero).

```
For Each c in Worksheets("Sheet1").Range("A1:D10")
   If c.Value < .001 Then
      c.Value = 0
   End If
Next c
```

This example loops on the range named "TestRange" and displays the number of empty cells in the range.

```
numBlanks = 0
For Each c In Range("TestRange")
   If c.Value = "" Then
      numBlanks = numBlanks + 1
   End If
Next c
MsgBox "There are " & numBlanks & " empty cells in this range"
```

This example sets the font style in cells A1:C5 on Sheet1 to italic. The example uses Syntax 2 of the **Range** property.

```
Worksheets("Sheet1").Range(Cells(1, 1), Cells(5, 3)).Font.Italic = True
```

RecentFiles Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRecentFilesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRecentFilesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRecentFilesA"}

Returns a **RecentFiles** collection that represents the list of recently used files.

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

RecentFiles Property Example

This example sets the maximum number of files in the list of recently used files to 6.

```
Application.RecentFiles.Maximum = 6
```

Maximum Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMaximumC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMaximumX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMaximumA"}

Returns or sets the maximum number of files in the list of recently used files. Can be a value from 0 (zero) through 9. Read/write **Long**.

Maximum Property Example

This example sets the maximum number of files in the list of recently used files to 6.

```
Application.RecentFiles.Maximum = 6
```

CustomViews Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCustomViewsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCustomViewsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCustomViewsA"}

Returns a **CustomViews** collection that represents all the custom views for the workbook.

For more information about returning a single object from a collection, see Returning an Object from a Collection.

CustomViews Property Example

This example creates a new custom view named "Summary" in the active workbook.

```
ActiveWorkbook.CustomViews.Add "Summary", True, True
```

View Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproViewC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproViewX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproViewA"}

Returns or sets the view showing in the window. Can be either of the following **XlWindowView** constants: **xlNormalView** or **xlPageBreakPreview**. Read/write **Long**.

View Property Example

This example switches the view in the active window to page break preview.

```
ActiveWindow.View = xlPageBreakPreview
```

PrintSettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPrintSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPrintSettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPrintSettingsA"}

True if print settings are included in the custom view. Read-only **Boolean**.

PrintSettings Property Example

This example creates a list of the custom views in the active workbook and their print settings and row and column settings.

```
With Worksheets(1)
    .Cells(1,1).Value = "Name"
    .Cells(1,2).Value = "Print Settings"
    .Cells(1,3).Value = "RowColSettings"
    rw = 0
    For Each v In ActiveWorkbook.CustomViews
       rw = rw + 1
       .Cells(rw, 1).Value = v.Name
       .Cells(rw, 2).Value = v.PrintSettings
       .Cells(rw, 3).Value = v.RowColSettings
    Next
End With
```

DragOff Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDragOffC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDragOffX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDragOffA"}

Drags a page break out of the print area.

Syntax

expression.**DragOff(*Direction, RegionIndex*)**

expression Required. An expression that returns an **HPageBreak** or **VPageBreak** object.

Direction Required **Long**. The direction in which the page break is dragged. Can be one of the following **XlDirection** constants: **xlDown**, **xlToLeft**, **xlToRight**, or **xlUp**.

RegionIndex Required **Long**. The print-area region index for the page break (the region where the mouse pointer is located when the mouse button is pressed if the user drags the page break). If the print area is contiguous, there's only one print region. If the print area is discontiguous, there's more than one print region.

Remarks

This method exists primarily for the macro recorder. You can use the **Delete** method to delete a page break in Visual Basic.

DragOff Method Example

This example deletes vertical page break one from the active sheet by dragging it off the right edge of print region one.

```
ActiveSheet.VPageBreaks(1).DragOff xlToRight, 1
```

Add Method (CustomViews Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddCustomViewsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddCustomViewsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddCustomViewsObjA"}

Creates a new custom view. Returns a **CustomView** object that represents the new view.

Syntax

expression.**Add(*ViewName, PrintSettings, RowColSettings*)**

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

ViewName Required **String**. The name of the new view.

PrintSettings Optional **Variant**. **True** to include print settings in the custom view.

RowColSettings Optional **Variant**. **True** to include settings for hidden rows and columns (including filter information) in the custom view.

Add Method (CustomViews Object) Example

This example creates a new custom view named "Summary" in the active workbook.

```
ActiveWorkbook.CustomViews.Add "Summary", True, True
```

RowColSettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowColSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowColSettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowColSettingsA"}

True if the custom view includes settings for hidden rows and columns (including filter information). Read-only **Boolean**.

RowColSettings Property Example

This example creates a list of the custom views in the active workbook and their print settings and row and column settings.

```
With Worksheets(1)
    .Cells(1,1).Value = "Name"
    .Cells(1,2).Value = "Print Settings"
    .Cells(1,3).Value = "RowColSettings"
    rw = 0
    For Each v In ActiveWorkbook.CustomViews
       rw = rw + 1
       .Cells(rw, 1).Value = v.Name
       .Cells(rw, 2).Value = v.PrintSettings
       .Cells(rw, 3).Value = v.RowColSettings
    Next
End With
```

AddToFavorites Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddToFavoritesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddToFavoritesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddToFavoritesA"}

Adds a shortcut to the workbook or hyperlink to the Favorites folder.

Syntax

expression.**AddToFavorites**

expression Required. An expression that returns a **Workbook** or **Hyperlink** object.

AddToFavorites Method Example

This example adds a shortcut to the active workbook to the Favorites folder.

```
ActiveWorkbook.AddToFavorites
```

Follow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFollowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFollowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFollowA"}

Displays a cached document, if it's already been downloaded. Otherwise, this method resolves the hyperlink, downloads the target document, and displays the document in the appropriate application.

Syntax

expression.**Follow(*NewWindow, AddHistory, ExtraInfo, Method, HeaderInfo*)**

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

NewWindow Optional **Variant**. **True** to display the target application in a new window. The default value is **False**.

AddHistory Optional **Variant**. Not used. Reserved for future use.

ExtraInfo Optional **Variant**. A **String** or byte array that specifies additional information for HTTP to use to resolve the hyperlink. For example, you can use ***ExtraInfo*** to specify the coordinates of an image map, the contents of a form, or a FAT file name.

Method Optional **Variant**. Specifies the way ***ExtraInfo*** is attached. Can be one of the following **MsoExtraInfoMethod** constants.

Constant	Description
msoMethodGet	***ExtraInfo*** is a **String** that's appended to the address.
msoMethodPost	***ExtraInfo*** is posted as a **String** or byte array.

HeaderInfo Optional **Variant**. A **String** that specifies header information for the HTTP request. The defaut value is an empty string.

Follow Method Example

This example loads the document attached to the hyperlink on shape one on worksheet one.

```
Worksheets(1).Shapes(1).Hyperlink.Follow NewWindow:=True
```

FollowHyperlink Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthFollowHyperlinkC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFollowHyperlinkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFollowHyperlinkA"}

Displays a cached document, if it's already been downloaded. Otherwise, this method resolves the hyperlink, downloads the target document, and displays the document in the appropriate application.

Syntax

expression.**FollowHyperlink(*Address, SubAddress, NewWindow, AddHistory, ExtraInfo, Method, HeaderInfo*)**

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

Address Required **String**. The address of the target document.

SubAddress Optional **Variant**. The location within the target document. The default value is the empty string.

NewWindow Optional **Variant**. **True** to display the target application in a new window. The default value is **False**.

AddHistory Optional **Variant**. Not used. Reserved for future use.

ExtraInfo Optional **Variant**. A **String** or byte array that specifies additional information for HTTP to use to resolve the hyperlink. For example, you can use ***ExtraInfo*** to specify the coordinates of an image map, the contents of a form, or a FAT file name.

Method Optional **Variant**. Specifies the way ***ExtraInfo*** is attached. Can be one of the following **MsoExtraInfoMethod** constants.

Constant	Description
msoMethodGet	***ExtraInfo*** is a **String** that's appended to the address.
msoMethodPost	***ExtraInfo*** is posted as a **String** or byte array.
msoMethodPostFile	***ExtraInfo*** specifies a FAT file name; the file content is posted.

HeaderInfo Optional **Variant**. A **String** that specifies header information for the HTTP request. The defaut value is an empty string.

FollowHyperlink Method Example

This example loads the document at www.gohere.com in a new window and adds it to the History folder.

```
ActiveWorkbook.FollowHyperlink Address:="http://www.gohere.com", _
   NewWindow:=True
```

Hyperlink Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHyperlinkC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHyperlinkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHyperlinkA"}

Returns a **Hyperlink** object that represents the hyperlink for the shape.

Hyperlink Property Example

This example loads the document attached to the hyperlink on shape one.

```
Worksheets(1).Shapes(1).Hyperlink.Follow NewWindow:=True
```

Hyperlinks Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHyperlinksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHyperlinksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHyperlinksA"}

Returns a **Hyperlinks** collection that represents the hyperlinks for the range or worksheet.

For more information about returning an object from a collection, see Returning an Object from a Collection.

Hyperlinks Property Example

This example checks to see whether any of the hyperlinks on worksheet one contain the word "Microsoft."

```
For Each h in Worksheets(1).Hyperlinks
   If Instr(h.Name, "Microsoft") <> 0 Then h.Follow
Next
```

Reload Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthReloadC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthReloadX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthReloadA"}

Reloads a cached workbook by resolving the hyperlink to the workbook and downloading it.

Syntax

expression.**Reload**

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

Reload Method Example

This example reloads the active workbook.

```
ActiveWorkbook.Reload
```

SubAddress Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSubAddressC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSubAddressX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSubAddressA"}

Returns or sets the location within the document associated with the hyperlink. Read/write **String**.

SubAddress Property Example

This example topic adds a range location to the hyperlink for shape one.

```
Worksheets(1).Shapes(1).Hyperlink.SubAddress = "A1:B10"
```

AddComment Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddCommentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddCommentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddCommentA"}

Adds a comment to the range.

Syntax

expression.**AddComment(*Text*)**

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

Text Optional **Variant**. The comment text.

AddComment Method Example

This example adds a comment to cell E5 on worksheet one.

```
Worksheets(1).Range("E5").AddComment "Current Sales"
```

AutoLoad Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoLoadC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoLoadX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoLoadA"}

True if the OLE object is automatically loaded when the workbook that contains it is opened. Read/write **Boolean**.

Remarks

This property is ignored by ActiveX controls. ActiveX controls are always loaded when a workbook is opened.

For most OLE object types, this property shouldn't be set to **True**. By default, the **AutoLoad** property is set to **False** for new OLE objects; this saves time and memory when Microsoft Excel is loading workbooks. The benefit of automatically loading OLE objects is that, for objects that represent volatile data, links to source data can be reestablished immediately and the objects can be rendered again, if necessary.

AutoLoad Property Example

This example sets the **AutoLoad** property for OLE object one on the active sheet.

```
ActiveSheet.OLEObjects(1).AutoLoad = False
```

CorrectCapsLock Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCorrectCapsLockC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCorrectCapsLockX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCorrectCapsLockA"}

True if Microsoft Excel automatically corrects accidental use of the CAPS LOCK key. Read/write **Boolean**.

CorrectCapsLock Property Example

This example enables Microsoft Excel to automatically correct accidental use of the CAPS LOCK key.

```
Application.AutoCorrect.CorrectCapsLock = True
```

CorrectSentenceCap Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCorrectSentenceCapC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCorrectSentenceCapX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCorrectSentenceCapA"}

True if Microsoft Excel automatically corrects sentence (first word) capitalization. Read/write **Boolean**.

CorrectSentenceCap Property Example

This example enables Microsoft Excel to automatically correct sentence capitalization.

```
Application.AutoCorrect.CorrectSentenceCap = True
```

DefaultSaveFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDefaultSaveFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDefaultSaveFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDefaultSaveFormatA"}

Returns or sets the default format for saving files. For a list of valid constants, see the **FileFormat** property. Read/write **Long**.

DefaultSaveFormat Property Example

This example sets the default format for saving files.

```
Application.DefaultSaveFormat = xlExcel4Workbook
```

DisplayCommentIndicator Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDisplayCommentIndicatorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDisplayCommentIndicatorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDisplayCommentIndicatorA"}

Returns or sets the way cells display comments and indicators. Can be one of the following **XlCommentDisplayMode** constants: **xlNoIndicator**, **xlCommentIndicatorOnly**, or **xlCommentAndIndicator**. Read/write **Long**.

DisplayCommentIndicator Property Example

This example hides cell tips but retains comment indicators.

```
Application.DisplayCommentIndicator = xlCommentIndicatorOnly
```

EnableCalculation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableCalculationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableCalculationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableCalculationA"}

True if Microsoft Excel automatically recalculates the worksheet when necessary. **False** if the user must request a recalculation (Microsoft Excel never recalculates the sheet automatically). Read/write **Boolean**.

Remarks

When you change this property from **False** to **True**, Microsoft Excel recalculates the worksheet.

EnableCalculation Property Example

This example sets Microsoft Excel to not recalculate worksheet one automatically.

```
Worksheets(1).EnableCalculation = False
```

EnableSound Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableSoundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableSoundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableSoundA"}

True if sound is enabled for Microsoft Office. Read/write **Boolean**.

EnableSound Property Example

This example disables sound feedback.

```
Application.EnableSound = False
```

Export Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthExportC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthExportX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthExportA"}

Exports the chart in a graphic format.

Syntax

expression.**Export(*FileName, FilterName, Interactive*)**

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

FileName Required **String**. The name of the exported file.

FilterName Optional **Variant**. The language-independent name of the graphic filter as it appears in the registry.

Interactive Optional **Variant**. **True** to display the dialog box that contains the filter-specific options. If this argument is **False**, Microsoft Excel uses the default values for the filter. The default value is **False**.

Export Method Example

This example exports chart one as a GIF file.

```
Charts(1).Export _
   OutputFileName:="current_sales.gif", FilterName:="GIF"
```

FileFind Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFileFindC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFileFindX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFileFindA"}

Returns a **FileFind** object for use with file searches. This property is available only on the Macintosh.

FileSearch Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFileSearchC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFileSearchX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFileSearchA"}

Returns a **FileSearch** object for use with file searches. This property is available only in Microsoft Windows.

FileSearch Property Example

This example creates a **FoundFiles** object that represents all the Microsoft Excel workbooks in the My Documents folder.

```
With Application.FileSearch
    .LookIn = "c:\my documents"
    .FileType = msoFileTypeExcelWorkbooks
    .Execute
End With
```

PageRangeCells Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPageRangeCellsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPageRangeCellsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPageRangeCellsA"}

Returns a **Range** object that represents the cells in the PivotTable containing only the page fields and item drop-down lists.

PageRangeCells Property Example

This example selects only the PivotTable cells that contain page fields and item drop-down lists.

```
Worksheets(1).PivotTables(1).PageRangeCells.Select
```

GetData Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGetDataC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGetDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGetDataA"}

Syntax

expression.**GetData(*Name*)**

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

Name Required **String**. Describes a single cell in the PivotTable, using syntax similar to the **PivotSelect** method or the PivotTable references in calculated item formulas.

GetData Method Example

This example shows the sum of revenues for apples in January (Data field = Revenue, Product = Apples, Month = January).

```
Msgbox ActiveSheet.PivotTables(1) _
    .GetData("'Sum of Revenue' Apples January")
```

PivotTableSelection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPivotTableSelectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPivotTableSelectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPivotTableSelectionA"}

True if PivotTables use structured selection. Read/write **Boolean**.

PivotTableSelection Property Example

This example enables structured selection mode and then sets PivotTable one to allow only data to be selected.

```
Application.PivotTableSelection = True
Worksheets(1).PivotTables(1).SelectionMode = xlDataOnly
```

ListFormulas Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthListFormulasC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthListFormulasX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthListFormulasA"}

Creates a list of calculated PivotTable items and fields on a separate worksheet.

Syntax

expression.**ListFormulas**

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

ListFormulas Method Example

This example creates a list of calculated items and fields for PivotTable one

```
Worksheets(1).PivotTables(1).ListFormulas
```

Has3DShading Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHas3DShadingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHas3DShadingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHas3DShadingA"}

True if the chart group has three-dimensional shading. Read/write **Boolean**.

Has3DShading Property Example

This example adds three-dimensional shading to chart group one on chart one.

```
Charts(1).ChartGroups(1).Has3DShading = True
```

HasAxis Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHasAxisC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHasAxisX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHasAxisA"}

Returns or sets which axes exist on the chart. Read/write **Variant**.

Syntax

expression.**HasAxis(*Index1, Index2*)**

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

Index1 Optional **Variant**. The axis type. Can be one of the following **XlAxisType** constants: **xlCategory**, **xlValue**, or **xlSeriesAxis**. Series axes apply only to 3-D charts.

Index2 Optional Variant. The axis group. Can be either of the following **XlAxisGroup** constants: **xlPrimary** or **xlSecondary**. 3-D charts have only one set of axes.

Remarks

Microsoft Excel may create or delete axes if you change the chart type or change the **AxisGroup** property.

HasAxis Property Example

This example turns on the primary value axis for Chart1.

```
Charts("Chart1").HasAxis(xlValue, xlPrimary) = True
```

ErrorTitle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproErrorTitleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproErrorTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproErrorTitleA"}

Returns or sets the title of the data-validation error dialog box. Read/write **String**.

ErrorTitle Property Example

This example adds data validation to cell E5.

```
With Range("e5").Validation
    .Add xlValidateWholeNumber, xlValidAlertInformation, xlBetween, "5",
"10"
    .InputTitle = "Integers"
    .ErrorTitle = "Integers"
    .InputMessage = "Enter an integer from five to ten"
    .ErrorMessage = "You must enter a number from five to ten"
End With
```

InputTitle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproInputTitleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproInputTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproInputTitleA"}

Returns or sets the title of the data-validation input dialog box. Read/write **String**.

InputTitle Property Example

This example turns on data validation for cell E5.

```
With Range("e5").Validation
    .Add xlValidateWholeNumber, xlValidAlertInformation, xlBetween, "5",
"10"
    .InputTitle = "Integers"
    .ErrorTitle = "Integers"
    .InputMessage = "Enter an integer from five to ten"
    .ErrorMessage = "You must enter a number from five to ten"
End With
```

IsAddin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIsAddinC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIsAddinX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIsAddinA"}

True if the workbook is running as an add-in. Read/write **Boolean**.

Remarks

When you set this property to **True**, the workbook has the following characteristics:

- You won't be prompted to save the workbook if changes are made while the workbook is open.
- The workbook window won't be visible.
- Any macros in the workbook won't be visible in the **Macro** dialog box (displayed by pointing to **Macro** on the **Tools** menu and clicking **Macros**).
- Macros in the workbook can still be run from the **Macro** dialog box even though they're not visible. In addition, macro names don't need to be qualified with the workbook name.
- Holding down the SHIFT key when you open the workbook has no effect.

IsAddin Property Example

This example runs a section of code if the workbook is an add-in.

```
If ThisWorkbook.IsAddin Then
   ' this code runs when the workbook is an add-in
End If
```

Next Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthNextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthNextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthNextA"}

Returns a **Comment** object that represents the next comment.

Syntax

expression.**Next**

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

Remarks

This method works only on one sheet. Using this method on the last comment on a sheet returns **Null** (not the next comment on the next sheet).

Next Method Example

This example hides the next comment.

```
Range("a1").Comment.Next.Visible = False
```

Previous Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPreviousC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPreviousX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPreviousA"}

Returns a **Comment** object that represents the previous comment.

Syntax

expression.**Previous**

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

Remarks

This method works only on one sheet. Using this method on the first comment on a sheet returns **Null** (not the last comment on the previous sheet).

Previous Method Example

This example hides the previous comment.

```
Range("a1").Comment.Previous.Visible = False
```

HighlightChangesOnScreen Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproHighlightChangesOnScreenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHighlightChangesOnScreenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHighlightChangesOnScreenA"}

True if changes to the shared workbook are highlighted on-screen. Read/write **Boolean**.

HighlightChangesOnScreen Property Example

This example highlights changes to the shared workbook.

```
ThisWorkbook.HighlightChangesOnScreen
```

ListChangesOnNewSheet Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproListChangesOnNewSheetC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproListChangesOnNewSheetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproListChangesOnNewSheetA"}

True if changes to the shared workbook are shown on a separate worksheet. Read/write **Boolean**.

ListChangesOnNewSheet Property Example

This example shows changes to the shared workbook on a separate worksheet.

```
With ActiveWorkbook
    .HighlightChangesOptions When:=xlSinceMyLastSave, Who:="Everyone"
    .ListChangesOnNewSheet = True
End With
```

HighlightChangesOptions Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthHighlightChangesOptionsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthHighlightChangesOptionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthHighlightChangesOptionsA"}

Controls how changes are shown in a shared workbook.

Syntax

expression.**HighlightChangesOptions(*When, Who, Where*)**

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

When Optional **Variant**. The changes that are shown. Can be one of the following **XlHighlightChangesTime** constants: **xlSinceMyLastSave**, **xlAllChanges**, or **xlNotYetReviewed**.

Who Optional **Variant**. The user or users whose changes are shown. Can be "Everyone," "Everyone but Me," or the name of one of the users of the shared workbook.

Where Optional **Variant**. An A1-style range reference that specifies the area to check for changes.

HighlightChangesOptions Method Example

This example shows changes to the shared workbook on a separate worksheet.

```
With ActiveWorkbook
    .HighlightChangesOptions When:=xlSinceMyLastSave, Who:="Everyone"
    .ListChangesOnNewSheet = True
End With
```

RefreshAll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRefreshAllC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRefreshAllX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRefreshAllA"}

Refreshes all external data ranges and PivotTables in the workbook.

Syntax

expression.**RefreshAll**

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

Remarks

The refresh order is undefined.

RefreshAll Method Example

This example refreshes all external data ranges and PivotTables in the workbook.

```
ThisWorkbook.RefreshAll
```

ReadingOrder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproReadingOrderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproReadingOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproReadingOrderA"}

This property isn't used in U.S. English Microsoft Excel.

ControlCharacters Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproControlCharactersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproControlCharactersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproControlCharactersA"}

This property isn't used in U.S. English Microsoft Excel.

CursorMovement Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCursorMovementC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCursorMovementX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCursorMovementA"}

This property isn't used in U.S. English Microsoft Excel.

DefaultSheetDirection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDefaultSheetDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDefaultSheetDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDefaultSheetDirectionA"}

This property isn't used in U.S. English Microsoft Excel.

IMEMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproIMEModeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproIMEModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproIMEModeA"}

This property isn't used in U.S. English Microsoft Excel.

UILanguage Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproUILanguageC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproUILanguageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproUILanguageA"}

This property isn't used in U.S. English Microsoft Excel.

Add Method (OLEObjects Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddOLEObjectsObjC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddOLEObjectsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddOLEObjectsObjA "}

Adds a new OLE object to a sheet. Returns an **OLEObject** object.

Syntax

expression.**Add(*ClassType, FileName, Link, DisplayAsIcon, IconFileName, IconIndex, IconLabel, Left, Top, Width, Height*)**

expression Required. An expression that returns an **OLEObjects** collection.

ClassType Optional **Variant**. (you must specify either ***ClassType*** or ***FileName***). A string that contains the programmatic identifier for the object to be created. If ***ClassType*** is specified, ***FileName*** and ***Link*** are ignored. For more information about programmatic identifiers, see OLE Programmatic Identifiers.

FileName Optional **Variant**. (you must specify either ***ClassType*** or ***FileName***). A string that specifies the file to be used to create the OLE object.

Link Optional **Variant**. **True** to have the new OLE object based on ***FileName*** be linked to that file. If the object isn't linked, the object is created as a copy of the file. The default value is **False**.

DisplayAsIcon Optional **Variant**. **True** to display the new OLE object either as an icon or as its regular picture. If this argument is **True**, ***IconFileName*** and ***IconIndex*** can be used to specify an icon.

IconFileName Optional **Variant**. A string that specifies the file that contains the icon to be displayed. This argument is used only if ***DisplayAsIcon*** is **True**. If this argument isn't specified or the file contains no icons, the default icon for the OLE class is used.

IconIndex Optional **Variant**. The number of the icon in the icon file. This is used only if ***DisplayAsIcon*** is **True** and ***IconFileName*** refers to a valid file that contains icons. If an icon with the given index number doesn't exist in the file specified by ***IconFileName***, the first icon in the file is used.

IconLabel Optional **Variant**. A string that specifies a label to display beneath the icon. This is used only if ***DisplayAsIcon*** is **True**. If this argument is omitted or is an empty string (""), no caption is displayed.

Left, ***Top*** Optional **Variant**. The initial coordinates of the new object, in points, relative to the upper-left corner of cell A1 on a worksheet, or to the upper-left corner of a chart.

Width, ***Height*** Optional **Variant**. The initial size of the new object, in points.

Add Method (OLEObjects Collection) Example

This example creates a new Microsoft Word OLE object on Sheet1.

```
ActiveWorkbook.Worksheets("Sheet1").OLEObjects.Add _
   ClassType:="Word.Document"
```

This example adds a command button to sheet one.

```
Worksheets(1).OLEObjects.Add ClassType:="Forms.CommandButton.1", _
   Link:=False, DisplayAsIcon:=False, Left:=40, Top:=40, _
   Width:=150, Height:=10
```

ProgId Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproProgIdC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproProgIdX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproProgIdA"}

Returns the programmatic identifiers for the object. Read-only **String**.

For more information about programmatic identifiers, see OLE Programmatic Identifiers.

ProgId Property Example

This example creates a list of the programmatic identifiers for the OLE objects on worksheet one.

```
rw = 0
For Each o in Worksheets(1).OLEObjects
   With Worksheets(2)
      rw = rw + 1
      .cells(rw, 1).Value = o.ProgId
   End With
Next
```

SetLinkOnData Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetLinkOnDataC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetLinkOnDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetLinkOnDataA"}

Sets the name of a procedure that runs whenever a DDE link is updated.

Syntax

expression.**SetLinkOnData(*Name, Procedure*)**

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

Name Required **String**. The name of the DDE/OLE link, as returned from the **LinkSources** method.

Procedure Required **String**. The name of the procedure to be run when the link is updated. This can be either a Microsoft Excel 4.0 macro or a Visual Basic procedure. Set this argument to an empty string ("") to indicate that no procedure should run when the link is updated.

SetLinkOnData Method Example

This example sets the name of the procedure that runs whenever the DDE link is updated.

```
ActiveWorkbook.SetLinkOnData "WinWord|'C:\MSGFILE.DOC'!DDE_LINK1", _
    "my_Link_Update_Macro"
```

SetSourceData Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetSourceDataC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetSourceDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetSourceDataA"}

Sets the source data range for the chart.

Syntax

expression.**SetSourceData(*Source, PlotBy*)**

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

Source Required **Range**. The range that contains the source data.

PlotBy Optional **Variant**. Specifies the way the data is to be plotted. Can be either of the following **XlRowCol** constants: **xlColumns** or **xlRows**.

SetSourceData Method Example

This example sets the source data range for chart one.

```
Charts(1).SetSourceData Source:=Sheets(1).Range("a1:a10"), _
    PlotBy:=xlColumns
```

TemplateRemoveExtData Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTemplateRemoveExtDataC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTemplateRemoveExtDataX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTemplateRemoveExtDataA"}

True if external data references are removed when the workbook is saved as a template. Read/write **Boolean**.

TemplateRemoveExtData Property Example

This example saves the workbook as a template that contains no external data.

```
With ThisWorkbook
    .TemplateRemoveExtData = True
    .SaveAs "current", xlTemplate
    .TemplateRemoveExtData = False
End With
```

Text Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthTextA"}

Sets comment text.

Syntax

expression.**Text(*Text, Start, Overwrite*)**

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

Text Optional **Variant**. The text to be added.

Start Optional **Variant**. The character number where the added text will be placed. If this argument is omitted, any existing text in the comment is deleted.

Overwrite Optional **Variant**. **True** to overwrite the existing text. The default value is **False** (text is inserted).

Text Method Example

This example adds a comment to cell E5 on sheet one.

```
With Worksheets(1).Range("e5").AddComment
    .Visible = False
    .Text "reviewed on " & Date
End With
```

Select Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSelectC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSelectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSelectA "}

Selects the object.

Syntax

expression.**Select(*Replace*)**

expression Required. An expression that returns an object in the Applies To list.

Replace Optional **Variant** (used only with sheets). **True** to replace the current selection with the specified object. **False** to extend the current selection to include any previously selected objects and the specified object.

Remarks

To select a cell or a range of cells, use the **Select** method. To make a single cell the active cell, use the **Activate** method.

Select Method Example

This example selects cells A1:B3 on Sheet1.

```
Worksheets("Sheet1").Activate
Range("A1:B3").Select
```

ResetColors Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthResetColorsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthResetColorsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthResetColorsA"}

Resets the color palette to the default colors.

Syntax

expression.**ResetColors**

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

ResetColors Method Example

This example resets the color palette in the active workbook.

```
ActiveWorkbook.ResetColors
```

BackColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBackColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBackColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBackColorA"}

ChartFillFormat object: Returns a **ChartColorFormat** object that represents the specified fill background color. Read-only.

All other objects: Returns a **ColorFormat** object that represents the specified fill background color. Read-only.

BackColor Property Example

This example sets the foreground color, background color, and gradient for the chart area fill on chart one.

```
With Charts(1).ChartArea.Fill
    .Visible = True
    .ForeColor.SchemeColor = 15
    .BackColor.SchemeColor = 17
    .TwoColorGradient msoGradientHorizontal, 1
End With
```

ForeColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproForeColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproForeColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproForeColorA"}

ChartFillFormat object: Returns a **ChartColorFormat** object that represents the specified fill foreground or solid color. Read-only.

All other objects: Returns a **ColorFormat** object that represents the specified fill foreground or solid color. Read-only.

ForeColor Property Example

This example sets the foreground color, background color, and gradient for the chart area fill on chart one.

```
With Charts(1).ChartArea.Fill
    .Visible = True
    .ForeColor.SchemeColor = 15
    .BackColor.SchemeColor = 17
    .TwoColorGradient msoGradientHorizontal, 1
End With
```

GradientColorType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGradientColorTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGradientColorTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGradientColorTypeA"}

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**.

GradientColorType Property Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillGradient And _
        c1f.GradientColorType = msoGradientOneColor Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        .OneColorGradient c1f.GradientStyle, _
            c1f.GradientVariant, c1f.GradientDegree
    End With
End If
```

GradientDegree Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGradientDegreeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGradientDegreeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGradientDegreeA"}

Returns the gradient degree of the specified one-color shaded fill as a floating-point value from 0.0 (dark) through 1.0 (light). 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 sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillGradient And _
        c1f.GradientColorType = msoGradientOneColor Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        .OneColorGradient c1f.GradientStyle, _
            c1f.GradientVariant, c1f.GradientDegree
    End With
End If
```

GradientStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGradientStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGradientStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGradientStyleA"}

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**. The **msoGradientFromTitle** constant is not used in Microsoft Excel. Read-only **Long**.

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

GradientStyle Property Example

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillGradient And _
        c1f.GradientColorType = msoGradientOneColor Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        .OneColorGradient c1f.GradientStyle, _
            c1f.GradientVariant, c1f.GradientDegree
    End With
End If
```

GradientVariant Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGradientVariantC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGradientVariantX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGradientVariantA"}

Returns the shade variant for the specified fill as an integer value from 1 through 4. The values for this property correspond to the gradient variants (numbered from left to right and from top to bottom) on the **Gradient** tab in the **Fill Effects** dialog box. Read-only **Long**.

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

GradientVariant Property Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillGradient And _
        c1f.GradientColorType = msoGradientOneColor Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        .OneColorGradient c1f.GradientStyle, _
            c1f.GradientVariant, c1f.GradientDegree
    End With
End If
```

OneColorGradient Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOneColorGradientC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOneColorGradientX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOneColorGradientA"}

Sets the specified fill to a one-color gradient.

Syntax

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

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

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

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

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

OneColorGradient Method Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillGradient And _
        c1f.GradientColorType = msoGradientOneColor Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        .OneColorGradient c1f.GradientStyle, _
            c1f.GradientVariant, c1f.GradientDegree
    End With
End If
```

Patterned Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPatternedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPatternedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPatternedA"}

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 type. Can be one of the following **MsoPatternType** constants:

msoPattern10Percent	**msoPatternLargeConfetti**
msoPattern20Percent	**msoPatternLargeGrid**
msoPattern25Percent	**msoPatternLightDiamonds**
msoPattern30Percent	**msoPatternLightDownwardDiagonal**
msoPattern40Percent	**msoPatternLightHorizontal**
msoPattern50Percent	**msoPatternLightUpwardDiagonal**
msoPattern5Percent	**msoPatternLightVertical**
msoPattern60Percent	**msoPatternMixed**
msoPattern70Percent	**msoPatternNarrowHorizontal**
msoPattern75Percent	**msoPatternNarrowVertical**
msoPattern80Percent	**msoPatternOutlinedDiamonds**
msoPattern90Percent	**msoPatternPlaid**
msoPatternDarkDownwardDiagonal	**msoPatternShingles**
msoPatternDarkHorizontal	**msoPatternSmallCheckerBoard**
msoPatternDarkUpwardDiagonal	**msoPatternSmallConfetti**
msoPatternDarkVertical	**msoPatternSmallGrid**
msoPatternDashedDownwardDiagonal	**msoPatternSolidDiamonds**
msoPatternDashedHorizontal	**msoPatternSpheres**
msoPatternDashedUpwardDiagonal	**msoPatternTrellis**
msoPatternDashedVertical	**msoPatternWaves**
msoPatternDiagonalBrick	**msoPatternWavyLines**
msoPatternDivits	**msoPatternWeave**
msoPatternDottedGrid	**msoPatternWideDownwardDiagonal**
msoPatternHorizontalBrick	**msoPatternWideUpwardDiagonal**
msoPatternLargeCheckerBoard	

Patterned Method Example

This example sets the fill pattern for chart one.

```
With Charts(1).ChartArea.Fill
    .Patterned msoPatternDiagonalBrick
    .Visible = True
End With
```

PresetGradient Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPresetGradientC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPresetGradientX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPresetGradientA"}

Sets the specified fill to a preset gradient.

Syntax

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

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

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

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

PresetGradientType Required **Long**. The gradient type. 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

PresetGradient Method Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillGradient Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        .PresetGradient c1f.GradientStyle, _
            c1f.GradientVariant, c1f.PresetGradientType
    End With
End If
```

PresetGradientType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPresetGradientTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetGradientTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetGradientTypeA"}

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

Use the **PresetGradient** method to set the preset gradient type for the fill.

PresetGradientType Property Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillGradient Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        .PresetGradient c1f.GradientStyle, _
            c1f.GradientVariant, c1f.PresetGradientType
    End With
End If
```

PresetTextured Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPresetTexturedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPresetTexturedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPresetTexturedA"}

Sets the specified fill format to a preset texture.

Syntax

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

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

PresetTexture Required **Long**. The preset texture. Can be one of the following **MsoPresetTexture** values: **msoPresetTextureMixed**, **msoTextureBrownMarble**, **msoTextureCloth**, **msoTextureCork**, **msoTextureGranite**, **msoTextureGreenMarble**, **msoTextureMediumWood**, **msoTextureOak**, **msoTexturePaper**, **msoTextureSand**, **msoTextureWalnut**, **msoTextureWhiteMarble**, or **msoTextureWovenMat**.

PresetTextured Method Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillTextured Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        If c1f.TextureType = msoTexturePreset Then
            .PresetTextured c1f.PresetTexture
        Else
            .UserTextured c1f.TextureName
        End If
    End With
End If
```

PresetTexture Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPresetTextureC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetTextureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetTextureA"}

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

Can be one of the following **MsoPresetTexture** values:

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

Use the **PresetTextured** method to set the preset texture for the fill.

PresetTexture Property Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillTextured Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        If c1f.TextureType = msoTexturePreset Then
            .PresetTextured c1f.PresetTexture
        Else
            .UserTextured c1f.TextureName
        End If
    End With
End If
```

RGB Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRGBC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRGBX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRGBA"}

ChartColorFormat object: Returns the red-green-blue value of the specified color. Read-only **Long**.

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

RGB Property Example

This example sets the interior color of the range A1:A10 to the chart area foreground fill color on chart one.

```
Worksheets(1).Range("A1:A10").Interior.Color = _
   Charts(1).ChartArea.Fill.ForeColor.RGB
```

SchemeColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSchemeColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSchemeColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSchemeColorA"}

Returns or sets the color of a **Color** object as an index in the current color scheme. Read/write **Long**.

SchemeColor Property Example

This example sets the foreground color, background color, and gradient for the chart area fill on chart one.

```
With Charts(1).ChartArea.Fill
    .Visible = True
    .ForeColor.SchemeColor = 15
    .BackColor.SchemeColor = 17
    .TwoColorGradient msoGradientHorizontal, 1
End With
```

TextureName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTextureNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTextureNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTextureNameA"}

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

Use the **UserPicture** or **UserTextured** method to set the texture file for the fill.

TextureName Property Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillTextured Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        If c1f.TextureType = msoTexturePreset Then
            .PresetTextured c1f.PresetTexture
        Else
            .UserTextured c1f.TextureName
        End If
    End With
End If
```

TextureType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTextureTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTextureTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTextureTypeA"}

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

Use the **UserTextured** method to set the texture type for the fill.

TextureType Property Example

This example sets the fill format for chart two to the same style used for chart one.

```
Set c1f = Charts(1).ChartArea.Fill
If c1f.Type = msoFillTextured Then
    With Charts(2).ChartArea.Fill
        .Visible = True
        If c1f.TextureType = msoTexturePreset Then
            .PresetTextured c1f.PresetTexture
        Else
            .UserTextured c1f.TextureName
        End If
    End With
End If
```

Transparency Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTransparencyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransparencyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransparencyA"}

Returns or sets the degree of transparency of the specified fill as a value from 0.0 (opaque) through 1.0 (clear). Read/write **Double**.

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 worksheet one to semitransparent red. If the shape doesn't already have a shadow, this example adds one to it.

```
With Worksheets(1).Shapes(3).Shadow
    .Visible = True
    .ForeColor.RGB = RGB(255, 0, 0)
    .Transparency = 0.5
End With
```

TwoColorGradient Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthTwoColorGradientC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthTwoColorGradientX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthTwoColorGradientA"}

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**, **msoGradientHorizontal**, or **msoGradientVertical**.

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

TwoColorGradient Method Example

This example sets the foreground color, background color, and gradient for the chart area fill on chart one.

```
With Charts(1).ChartArea.Fill
    .Visible = True
    .ForeColor.SchemeColor = 15
    .BackColor.SchemeColor = 17
    .TwoColorGradient msoGradientHorizontal, 1
End With
```

UserPicture Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUserPictureC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUserPictureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUserPictureA"}

Fills the specified shape with an image.

Syntax

expression.**UserPicture(*PictureFile, PictureFormat, PictureStackUnit, PicturePlacement*)**

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

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

PictureFormat Required **Long**. The picture format. Can be one of the following **XlChartPictureType** constants: **xlStack**, **xlStackScale**, or **xlStretch**.

PictureStackUnit Required **Long**. The picture stack or scale unit (depends on the ***PictureFormat*** argument).

PicturePlacement Required **Long**. The picture placement. Can be one of the following **XlChartPicturePlacement** constants: **xlAllFaces**, **xlEnd**, **xlEndSides**, **xlFront**, **xlFrontEnd**, **xlFrontSides**, or **xlSides**.

UserPicture Method Example

This example sets the fill format for chart two.

```
Charts(2).ChartArea.Fill.UserPicture "brick.bmp
```

UserTextured Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUserTexturedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUserTexturedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUserTexturedA"}

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 sets the fill format for chart two.

```
Charts(2).ChartArea.Fill.UserTextured "brick.bmp"
```

ApplyCustomType Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthApplyCustomTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthApplyCustomTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthApplyCustomTypeA"}

Applies a standard or custom chart type to a chart or series.

Syntax

expression.**ApplyCustomType(*ChartType, TypeName*)**

expression Required. An expression that returns a **Chart** or **Series** object.

ChartType Required **Long**. A standard chart type (see the **ChartType** property for the list of available constants). For **Chart** objects, this argument can also be one of the following **XlChartGallery** constants: **xlBuiltIn**, **xlUserDefined**, or **xlAnyGallery**.

TypeName Optional **Variant** (used only with **Chart** objects). The name of the custom chart type if ***ChartType*** specifies a custom chart gallery.

ApplyCustomType Method Example

This example applies the "Line with Data Markers" chart type to chart one.

```
Charts(1).ApplyCustomType xlLineMarkers
```

AutoShowCount Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoShowCountC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoShowCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoShowCountA"}

Returns the number of top or bottom items that are automatically shown in the pivot field. Read-only **Long**.

AutoShowCount Property Example

This example displays a message box showing the AutoShow parameters for the Salesman field.

```
With Worksheets(1).PivotTables(1).PivotFields("salesman")
    If .AutoShowType = xlAutomatic Then
        r = .AutoShowRange
        If r = xlTop Then
            rn = "top"
        Else
            rn = "bottom"
        End If
        MsgBox "PivotTable is showing " & rn & " " & _
            .AutoShowCount & " items in " & .Name & _
            " field by " & .AutoShowField
    Else
        MsgBox "Pivot table is not using AutoShow for this field"
    End If
End With
```

AutoShowField Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoShowFieldC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoShowFieldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoShowFieldA"}

Returns the name of the data field used to determine the top or bottom items that are automatically shown in the pivot field. Read-only **String**.

AutoShowField Property Example

This example displays a message box showing the AutoShow parameters for the Salesman field.

```
With Worksheets(1).PivotTables(1).PivotFields("salesman")
    If .AutoShowType = xlAutomatic Then
        r = .AutoShowRange
        If r = xlTop Then
            rn = "top"
        Else
            rn = "bottom"
        End If
        MsgBox "PivotTable is showing " & rn & " " & _
            .AutoShowCount & " items in " & .Name & _
            " field by " & .AutoShowField
    Else
        MsgBox "Pivot table is not using AutoShow for this field"
    End If
End With
```

AutoShowRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoShowRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoShowRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoShowRangeA"}

Returns **xlTop** if the top items are shown automatically in the pivot field; returns **xlBottom** if the bottom items are shown. Read-only **Long**.

AutoShowRange Property Example

This example displays a message box showing the AutoShow parameters for the Salesman field.

```
With Worksheets(1).PivotTables(1).PivotFields("salesman")
    If .AutoShowType = xlAutomatic Then
        r = .AutoShowRange
        If r = xlTop Then
            rn = "top"
        Else
            rn = "bottom"
        End If
        MsgBox "PivotTable is showing " & rn & " " & _
            .AutoShowCount & " items in " & .Name & _
            " field by " & .AutoShowField
    Else
        MsgBox "Pivot table is not using AutoShow for this field"
    End If
End With
```

AutoShowType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoShowTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoShowTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoShowTypeA"}

Returns **xlAutomatic** if AutoShow is enabled for the pivot field; returns **xlManual** if AutoShow is disabled. Read-only **Long**.

AutoShowType Property Example

This example displays a message box showing the AutoShow parameters for the Salesman field.

```
With Worksheets(1).PivotTables(1).PivotFields("salesman")
    If .AutoShowType = xlAutomatic Then
        r = .AutoShowRange
        If r = xlTop Then
            rn = "top"
        Else
            rn = "bottom"
        End If
        MsgBox "PivotTable is showing " & rn & " " & _
            .AutoShowCount & " items in " & .Name & _
            " field by " & .AutoShowField
    Else
        MsgBox "Pivot table is not using AutoShow for this field"
    End If
End With
```

AutoSortField Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoSortFieldC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoSortFieldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoSortFieldA"}

Returns the name of the data field used to sort the pivot field automatically. Read-only **String**.

AutoSortField Property Example

This example displays a message box showing the AutoSort parameters for the Product field.

```
With Worksheets(1).PivotTables(1).PivotFields("product")
    Select Case .AutoSortOrder
        Case xlManual
            aso = "manual"
        Case xlAscending
            aso = "ascending"
        Case xlDescending
            aso = "descending"
    End Select
    MsgBox " sorted in " & aso & _
        " order by " & .AutoSortField
End With
```

AutoSortOrder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoSortOrderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoSortOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoSortOrderA"}

Returns the order used to sort the pivot field automatically. Can be one of the following **XlSortOrder** constants: **xlAscending** or **xlDescending** (or **xlManual** if automatic sorting is disabled). Read-only **Long**.

AutoSortOrder Property Example

This example displays a message box showing the AutoSort parameters for the Product field.

```
With Worksheets(1).PivotTables(1).PivotFields("product")
    Select Case .AutoSortOrder
        Case xlManual
            aso = "manual"
        Case xlAscending
            aso = "ascending"
        Case xlDescending
            aso = "descending"
    End Select
    MsgBox " sorted in " & aso & _
        " order by " & .AutoSortField
End With
```

ClearComments Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthClearCommentsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthClearCommentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthClearCommentsA"}

Clears all cell comments from the specified range.

Syntax

expression.**ClearComments**

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

ClearComments Method Example

This example clears all comments from cell E5.

```
Worksheets(1).Range("e5").ClearComments
```

KeepChangeHistory Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproKeepChangeHistoryC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproKeepChangeHistoryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproKeepChangeHistoryA"}

True if change tracking is enabled for the shared workbook. Read/write **Boolean**.

KeepChangeHistory Property Example

This example sets the number of days shown in the change history for the active workbook if change tracking is enabled.

```
With ActiveWorkbook
   If .KeepChangeHistory Then
      .ChangeHistoryDuration = 7
   End If
End With
```

RollZoom Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRollZoomC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRollZoomX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRollZoomA"}

True if the IntelliMouse zooms instead of scrolling. Read/write **Boolean**.

RollZoom Property Example

This example enables the IntelliMouse to zoom instead of scroll.

```
Application.RollZoom = True
```

UnprotectSharing Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUnprotectSharingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUnprotectSharingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUnprotectSharingA"}

Turns off protection for sharing and saves the workbook.

Syntax

expression.**UnprotectSharing(*SharingPassword*)**

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

SharingPassword Optional **Variant**. The workbook password.

UnprotectSharing Method Example

This example turns off protection for sharing and saves the active workbook.

```
ActiveWorkbook.UnprotectSharing Password:="drowssap"
```

Value2 Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproValue2C"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproValue2X":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproValue2A"}

Returns or sets the cell value. Read/write **Variant**.

Remarks

The only difference between this property and the **Value** property is that the **Value2** property doesn't use the **Currency** and **Date** data types. You can return values formatted with these data types as floating-point numbers by using the **Double** data type.

Value2 Property Example

This example uses the **Value2** property to add the values of two cells.

```
Range("a1").Value2 = Range("b1").Value2 + Range("c1").Value2
```

WhichAddress Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproWhichAddressC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproWhichAddressX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproWhichAddressA"}

Returns or sets the PowerTalk address type. Available only in Microsoft Excel for the Macintosh, with the PowerTalk mail system extension installed. Read/write **Variant**.

Type Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTypeC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTypeA "}

Returns or sets the object type, as shown in the following table.

Object	Type
Axis	Axis type. Can be one of the following **XlAxisType** constants: **xlCategory**, **xlSeriesAxis**, or **xlValue**. Read/write **Long**.
Color	Color type. Can be one of the following **MsoColorType** constants: **msoColorTypeMixed**, **msoColorTypeRGB**, or **msoColorTypeScheme**. Read-only **Long**.
ConnectorFormat	Connector type. Can be one of the following **MsoConnectorType** constants: **msoConnectorCurve**, **msoConnectorElbow**, **msoConnectorStraight**, or **msoConnectorTypeMixed**. Read/write **Long**.
DataLabel, DataLabels	Data label type. Can be one of the following **XlDataLabelsType** constants: **xlDataLabelsShowBubbleSizes**, **xlDataLabelsShowLabel**, **xlDataLabelsShowLabelAndPercent**, **xlDataLabelsShowNone**, **xlDataLabelsShowPercent**, or **xlDataLabelsShowValue**. Read/write **Long**.
FillFormat	Fill type. Can be one of the following **MsoFillType** constants: **msoFillBackground**, **msoFillGradient**, **msoFillMixed**, **msoFillPatterned**, **msoFillPicture**, **msoFillSolid**, or **msoFillTextured**. The **msoFillBackground** constant is not used in Microsoft Excel. Read-only **Long**.
FormatCondition	Conditional format type. Can be one of the following **xlFormatConditionType** constants: **xlCellValue** or **xlExpression**. Read-only **Long**.
HPageBreak, VPageBreak	Page break type. Can be one of the following **XlPageBreak** constants: **xlPageBreakAutomatic** or **xlPageBreakManual**. Read/write **Long**.
Hyperlink	Hyperlink type (what the hyperlink is associated with). Can be one of the following **MsoHyperlinkType** constants: **msoHyperlinkInlineShape**, **msoHyperlinkRange**, or **msoHyperlinkShape**. Read-only **Long**.
Parameter	Parameter type. Can be one of the following **XlParameterType** constants: **xlConstant**, **xlPrompt**, or **xlRange**. Read/write **Long**.
Shape, **ShapeRange**	Shape type. Can be one of the following **MsoShapeType** constants: **msoAutoShape**, **msoCallout**, **msoChart**, **msoComment**, **msoEmbeddedOLEObject**, **msoFormControl**, **msoFreeform**, **msoGroup**, **msoLine**, **msoLinkedOLEObject**, **msoLinkedPicture**,

	msoMedia, **msoOLEControlObject**, **msoPicture**, **msoPlaceholder**, **msoShapeTypeMixed**, or **msoTextEffect**. Read-only **Long**. In Microsoft Excel, this property cannot be **msoMedia** or **msoPlaceholder** (these constants are used with shapes in other Microsoft Office applications)..
Trendline	Trendline type. Can be one of the following **XlTrendlineType** constants: **xlExponential**, **xlLinear**, **xlLogarithmic**, **xlMovingAvg**, **xlPolynomial**, or **xlPower**. Read/write **Long**.
Validation	Data validation type. Can be one of the following **XlDVType** constants: **xlValidateCustom**, **xlValidateDate**, **xlValidateDecimal**, **xlValidateInputOnly**, **xlValidateList**, **xlValidateTextLength**, **xlValidateTime**, or **xlValidateWholeNumber**. Read-only **Long**.
Window	Window type. Can be one of the following **XlWindowType** constants: **xlChartAsWindow**, **xlChartInPlace**, **xlClipboard**, **xlInfo**, or **xlWorkbook**. Read-only **Long**.
Worksheet	Worksheet type. Can be one of **xlWorksheet**, **xlExcel4MacroSheet**, or **xlExcel4IntlMacroSheet**. Read-only **Long**.

Type Property Example

This example changes the trendline type for the first series in embedded chart one on worksheet one. If the series has no trendline, this example fails.

```
Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection(1).Trendlines(1).Type = xlMovingAvg
```

ShapeRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShapeRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShapeRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShapeRangeA"}

Returns a **ShapeRange** object that represents the specified object or objects. Read-only.

ShapeRange Property Example

This example creates a shape range that represents the embedded charts on worksheet one.

```
Set sr = Worksheets(1).ChartObjects.ShapeRange
```

TextFrame Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTextFrameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTextFrameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTextFrameA"}

Returns a **TextFrame** object that contains the alignment and anchoring properties for the specified shape. Read-only.

TextFrame Property Example

This example causes text in the text frame in shape one to be justified. If shape one doesn't have a text frame, this example fails.

```
Worksheets(1).Shapes(1).TextFrame.HorizontalAlignment = xlHAlignJustify
```

VBE Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVBEC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVBEX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVBEA"}

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

VBE Property Example

This example changes the name of the active Visual Basic project.

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

VBProject Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVBProjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVBProjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVBProjectA"}

Returns a **VBProject** object that represents the Visual Basic project in the specified workbook. Read-only.

VBProject Property Example

This example changes the name of the Visual Basic project in the workbook.

```
ThisWorkbook.VBProject.Name = "TestProject"
```

LinkedCell Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLinkedCellC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLinkedCellX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLinkedCellA"}

Returns or sets the worksheet range linked to the control's value. If you place a value in the cell, the control takes this value. Likewise, if you change the value of the control, that value is also placed in the cell. Read/write **String**.

Remarks

You cannot use this property with multiselect list boxes.

LinkedCell Property Example

This example adds a check box to worksheet one and links the check box value to cell A1.

```
With Worksheets(1)
   Set cb = .Shapes.AddFormControl(xlCheckBox, 10, 10, 100, 10)
   cb.ControlFormat.LinkedCell = "A1"
End With
```

ListFillRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproListFillRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproListFillRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproListFillRangeA"}

Returns or sets the worksheet range used to fill the specified list box. Setting this property destroys any existing list in the list box. Read/write **String**.

Remarks

Microsoft Excel reads the contents of every cell in the range and inserts the cell values into the list box. The list tracks changes in the range's cells.

If the list in the list box was created with the **AddItem** method, this property returns an empty string ("").

ListFillRange Property Example

This example adds a list box to worksheet one and sets the fill range for the list box.

```
With Worksheets(1)
   Set lb = .Shapes.AddFormControl(xlListBox, 100, 10, 100, 100)
   lb.ControlFormat.ListFillRange = "A1:A10"
End With
```

SourceName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSourceNameC "} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSourceNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSourceNameA "}

OLEObject, **OLEObjects** objects: Return or set the specified object's link source name. Read/write **String**.

PivotField, PivotItem objects: Return the specified object's name, as it appears in the original source data for the PivotTable. This might be different from the current item name if the user renamed the item after creating the PivotTable. Read-only **String**.

SourceName Property Example

This example displays the original name (the name from the source database) of the item that contains the active cell.

```
Worksheets("Sheet1").Activate
ActiveSheet.PivotTables(1).PivotSelect "1998", xlDataAndLabel
MsgBox "The original item name is " & _
   ActiveCell.PivotItem.SourceName
```

AddFormControl Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddFormControlC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddFormControlX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddFormControlA"}

Creates a Microsoft Excel control. Returns a **Shape** object that represents the new control.

Syntax

expression.**AddFormControl(*Type, Left, Top, Width, Height*)**

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

Type Required **Long**. The Microsoft Excel control type. Can be one of the following **XlFormControl** constants: **xlButtonControl**, **xlCheckBox**, **xlDropDown**, **xlEditBox**, **xlGroupBox**, **xlLabel**, **xlListBox**, **xlOptionButton**, **xlScrollBar**, or **xlSpinner**. You cannot create an edit box on a worksheet.

Left, ***Top*** Required **Long**. The initial coordinates of the new object (in points) relative to the upper-left corner of cell A1 on a worksheet or to the upper-left corner of a chart.

Width, ***Height*** Required **Long**. The initial size of the new object, in points.

Remarks

Use the **AddOLEObject** method or the **Add** method of the **OLEObjects** collection to create an ActiveX control.

AddFormControl Method Example

This example adds a list box to worksheet one and sets the fill range for the list box.

```
With Worksheets(1)
   Set lb = .Shapes.AddFormControl(xlListBox, 100, 10, 100, 100)
   lb.ControlFormat.ListFillRange = "A1:A10"
End With
```

AddItem Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddItemC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddItemA"}

Adds an item to a list box or a combo box.

Syntax

expression.**AddItem(*Text, Index*)**

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

Text Required **String**. The text to be added

Index Optional **Variant**. The position of the new entry. If the list has fewer entries than the specified index, blank items from the end of the list are added to the specified position. If this argument is omitted, the item is appended to the existing list.

Remarks

Using this method clears any range specified by the **ListFillRange** property.

AddItem Method Example

This example creates a list box and fills it with integers from 1 to 10.

```
With Worksheets(1)
   Set lb = .Shapes.AddFormControl(xlListBox, 100, 10, 100, 100)
   For x = 1 To 10
      lb.ControlFormat.AddItem x
   Next
End With
```

List Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthListC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthListX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthListA"}

Returns or sets the text entries in the specified list box or a combo box, as an array of strings, or returns or sets a single text entry. An error occurs if there are no entries in the list.

Syntax

expression.**List(*Index*)**

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

Index Optional **Variant**. The index number of a single text entry to be set or returned. If this argument is omitted, the entire list is returned or set as an array of strings.

Remarks

Setting this property clears any range specified by the **ListFillRange** property.

List Method Example

This example sets the entries in a list box on worksheet one. If `Shapes(2)` doesn't represent a list box, this example fails.

```
Worksheets(1).Shapes(2).ControlFormat.List = _
   Array("cogs", "widgets", "sprockets", "gizmos")
```

This example sets entry four in a list box on worksheet one. If `Shapes(2)` doesn't represent a list box, this example fails.

```
Worksheets(1).Shapes(2).ControlFormat.List(4) = "gadgets"
```

OnKey Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOnKeyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOnKeyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOnKeyA"}

Runs a specified procedure when a particular key or key combination is pressed.

Syntax

expression.**OnKey(*Key, Procedure*)**

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

Key Required **String**. A string indicating the key to be pressed.

Procedure Optional **Variant**. A string indicating the name of the procedure to be run. If ***Procedure*** is "" (empty text), nothing happens when ***Key*** is pressed. This form of **OnKey** changes the normal result of keystrokes in Microsoft Excel. If ***Procedure*** is omitted, ***Key*** reverts to its normal result in Microsoft Excel, and any special key assignments made with previous **OnKey** methods are cleared.

Remarks

The ***Key*** argument can specify any single key; any key combined with ALT, CTRL, or SHIFT, or any combination of these keys (in Windows); or COMMAND, CTRL, OPTION, or SHIFT, or any combination of these keys (on the Macintosh). Each key is represented by one or more characters, such as `"a"` for the character a, or `"{ENTER}"` for the ENTER key.

To specify characters that aren't displayed when you press the corresponding key (ENTER or TAB, for example), use the codes listed in the following table. Each code in the table represents one key on the keyboard.

Key	Code
BACKSPACE	`{BACKSPACE}` or `{BS}`
BREAK	`{BREAK}`
CAPS LOCK	`{CAPSLOCK}`
CLEAR	`{CLEAR}`
DELETE or DEL	`{DELETE}` or `{DEL}`
DOWN ARROW	`{DOWN}`
END	`{END}`
ENTER (numeric keypad)	`{ENTER}`
ENTER	~ (tilde)
ESC	`{ESCAPE}` or `{ESC}`
HELP	`{HELP}`
HOME	`{HOME}`
INS	`{INSERT}`
LEFT ARROW	`{LEFT}`
NUM LOCK	`{NUMLOCK}`
PAGE DOWN	`{PGDN}`
PAGE UP	`{PGUP}`
RETURN	`{RETURN}`
RIGHT ARROW	`{RIGHT}`
SCROLL LOCK	`{SCROLLLOCK}`
TAB	`{TAB}`
UP ARROW	`{UP}`

F1 through F15	{F1} through {F15}

In Windows, you can also specify keys combined with SHIFT and/or CTRL and/or ALT. On the Macintosh, you can also specify keys combined with SHIFT and/or CTRL and/or OPTION and/or COMMAND. To specify a key combined with another key or keys, use the following table.

To combine keys with	Precede the key code by
SHIFT	+ (plus sign)
CTRL	^ (caret)
ALT or OPTION	% (percent sign)
COMMAND	* (asterisk)

To assign a procedure to one of the special characters (+, ^, %, and so on), enclose the character in braces. For details, see the example.

OnKey Method Example

This example assigns "InsertProc" to the key sequence CTRL+PLUS SIGN and assigns "SpecialPrintProc" to the key sequence SHIFT+CTRL+RIGHT ARROW.

```
Application.OnKey "^{+}", "InsertProc"
Application.OnKey "+^{RIGHT}", "SpecialPrintProc"
```

This example returns SHIFT+CTRL+RIGHT ARROW to its normal meaning.

```
Application.OnKey "+^{RIGHT}"
```

This example disables the SHIFT+CTRL+RIGHT ARROW key sequence.

```
Application.OnKey "+^{RIGHT}", ""
```

OnRepeat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOnRepeatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOnRepeatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOnRepeatA"}

Sets the **Repeat** menu item and the name of the procedure that will run if you choose the **Repeat** command (**Edit** menu) after running the procedure that sets this property.

Syntax

expression.**OnRepeat(*Text, Procedure*)**

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

Text Required **String**. The text that appears with the **Repeat** command (**Edit** menu).

Procedure Required **String**. The name of the procedure that will be run when you choose the **Repeat** command (**Edit** menu).

Remarks

If a procedure doesn't use the **OnRepeat** method, the **Repeat** command repeats procedure that was run most recently.

The procedure must use the **OnRepeat** and **OnUndo** methods last, to prevent the repeat and undo procedures from being overwritten by subsequent actions in the procedure.

OnRepeat Method Example

This example sets the repeat and undo procedures.

```
Application.OnRepeat "Repeat VB Procedure", _
   "Book1.xls!My_Repeat_Sub"
Application.OnUndo "Undo VB Procedure", _
   "Book1.xls!My_Undo_Sub"
```

OnTime Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOnTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOnTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOnTimeA"}

Schedules a procedure to be run at a specified time in the future (either at a specific time of day or after a specific amount of time has passed).

Syntax

expression.**OnTime(*EarliestTime, Procedure, LatestTime, Schedule*)**

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

EarliestTime Required **Variant**. The time when you want this procedure to be run.

Procedure Required **String**. The name of the procedure to be run.

LatestTime Optional **Variant**. The latest time at which the procedure can be run. For example, if ***LatestTime*** is set to ***EarliestTime*** + 30 and Microsoft Excel is not in Ready, Copy, Cut, or Find mode at ***EarliestTime*** because another procedure is running, Microsoft Excel will wait 30 seconds for the first procedure to complete. If Microsoft Excel is not in Ready mode within 30 seconds, the procedure won't be run. If this argument is omitted, Microsoft Excel will wait until the procedure can be run.

Schedule Optional **Variant**. **True** to schedule a new **OnTime** procedure. **False** to clear a previously set procedure. The default value is **True**.

Remarks

Use `Now + TimeValue(time)` to schedule something to be run when a specific amount of time (counting from now) has elapsed. Use `TimeValue(time)` to schedule something to be run a specific time.

OnTime Method Example

This example runs my_Procedure 15 seconds from now.

```
Application.OnTime Now + TimeValue("00:00:15"), "my_Procedure"
```

This example runs my_Procedure at 5 P.M.

```
Application.OnTime TimeValue("17:00:00"), "my_Procedure"
```

This example cancels the **OnTime** setting from the previous example.

```
Application.OnTime EarliestTime:=TimeValue("17:00:00"), _
   Procedure:="my_Procedure", Schedule:=False
```

OnUndo Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthOnUndoC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthOnUndoX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthOnUndoA"}

Sets the text of the **Undo** and the name of the procedure that's run if you choose the **Undo** command (**Edit** menu) after running the procedure that sets this property.

Syntax

expression.**OnUndo(*Text, Procedure*)**

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

Text Required **String**. The text that appears with the **Undo** command (**Edit** menu).

Procedure Required **String**. The name of the procedure that's run when you choose the **Undo** command (**Edit** menu).

Remarks

If a procedure doesn't use the **OnUndo** method, the **Undo** command is disabled.

The procedure must use the **OnRepeat** and **OnUndo** methods last, to prevent the repeat and undo procedures from being overwritten by subsequent actions in the procedure.

OnUndo Method Example

This example sets the repeat and undo procedures.

```
Application.OnRepeat "Repeat VB Procedure", _
   "Book1.xls!My_Repeat_Sub"
Application.OnUndo "Undo VB Procedure", _
   "Book1.xls!My_Undo_Sub"
```

ProtectSharing Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthProtectSharingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthProtectSharingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthProtectSharingA"}

Saves the workbook and protects it for sharing.

Syntax

expression.**ProtectSharing(*Filename, Password, WriteResPassword, ReadOnlyRecommended, CreateBackup, SharingPassword*)**

expression An expression that returns a **Workbook** object.

Filename Optional **Variant**. A string indicating the name of the saved file. You can include a full path; if you don't, Microsoft Excel saves the file in the current folder.

Password Optional **Variant**. A case-sensitive string indicating the protection password to be given to the file. Should be no longer than 15 characters.

WriteResPassword Optional **Variant**. A string indicating the write-reservation password for this file. If a file is saved with the password and the password isn't supplied when the file is opened, the file is opened read-only.

ReadOnlyRecommended Optional **Variant**. **True** to display a message when the file is opened, recommending that the file be opened read-only.

CreateBackup Optional **Variant**. **True** to create a backup file.

SharingPassword Optional **Variant**. A string indicating the password to be used to protect the file for sharing.

ProtectSharing Method Example

This example saves workbook one and protects it for sharing.

```
Workbooks(1).ProtectForSharing Password:="drowssap", _
    SharingPassword:="gnirahs"
```

RemoveAllItems Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRemoveAllItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRemoveAllItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRemoveAllItemsA"}

Removes all entries from a Microsoft Excel list box or combo box. Use the **Clear** method to remove all items from an ActiveX list box or combo box.

Syntax

expression.**RemoveAllItems**

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

RemoveAllItems Method Example

This example removes all items from a list box. If `Shapes(2)` doesn't represent a list box, this example fails.

```
Worksheets(1).Shapes(2).ControlFormat.RemoveAllItems
```

RemoveItem Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRemoveItemC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRemoveItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRemoveItemA"}

Removes one or more items from a list box or combo box.

Syntax

expression.**RemoveItem(*Index, Count*)**

expression An expression that returns a **ControlFormat** object.

Index Required **Long**. The number of the first item to be removed. Valid values are from 1 to the number of items in the list (returned by the **ListCount** property).

Count Optional **Variant**. The number of items to be removed, starting at item ***Index***. If this argument is omitted, one item is removed. If ***Index*** + ***Count*** exceeds the number of items in the list, all items from ***Index*** through the end of the list are removed without an error.

Remarks

If the specified object has a fill range defined for it, this method fails.

Use the **RemoveAllItems** method to remove all entries from a Microsoft Excel list box or combo box. Use the **Clear** method to remove all items from an ActiveX list box or combo box.

RemoveItem Method Example

This example removes the selected item from a list box. If `Shapes(2)` doesn't represent a list box, this example fails.

```
Set lbcf = Worksheets(1).Shapes(2).ControlFormat
lbcf.RemoveItem lbcf.ListIndex
```

ChangeHistoryDuration Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproChangeHistoryDurationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproChangeHistoryDurationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproChangeHistoryDurationA"}

Returns or sets the number of days shown in the shared workbook's change history. Read/write **Long**.

Remarks

Any changes in the change history older than the setting for this property are removed when the workbook is closed.

ChangeHistoryDuration Property Example

This example sets the number of days shown in the change history for the active workbook if change tracking is enabled. Any changes in the change history older than the setting for this property are removed when the workbook is closed.

```
With ActiveWorkbook
   If .KeepChangeHistory Then
      .ChangeHistoryDuration = 7
   End If
End With
```

Characters Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCharactersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCharactersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCharactersA"}

Returns a **Characters** object that represents a range of characters within the object text. You can use the **Characters** object to format characters within a text string.

Syntax

expression.**Characters(*Start, Length*)**

expression Required. An expression that returns an object in the Applies To list.

Start Optional **Variant**. The first character to be returned. If this argument is either 1 or omitted, this method returns a range of characters starting with the first character.

Length Optional **Variant**. The number of characters to be returned. If this argument is omitted, this method returns the remainder of the string (everything after the ***Start*** character).

Remarks

The **Characters** object isn't a collection.

When applied to a **Range** object, this property fails if it's used with arguments and the cell doesn't contain a text value.

Characters Property Example

This example formats the third character in cell A1 on Sheet1 as bold.

```
With Worksheets("Sheet1").Range("A1")
    .Value = "abcdefg"
    .Characters(3, 1).Font.Bold = True
End With
```

CodeName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCodeNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCodeNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCodeNameA"}

Returns the code name for the object. Read-only **String**.

Remarks

The code name for an object can be used in place of an expression that returns the object. For example, if the code name for worksheet one is "Sheet1", the following expressions are identical:

```
Worksheets(1).Range("a1")
Sheet1.Range("a1")
```

It's possible for the sheet name to be different from the code name. When you create a sheet, the sheet name and code name are the same, but changing the sheet name doesn'tchange the code name, and changing the code name (using the **Properties** window in the Visual Basic Editor) doesn't change the sheet name.

CodeName Property Example

This example displays the code name for worksheet one.

```
MsgBox Worksheets(1).CodeName
```

ControlFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproControlFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproControlFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproControlFormatA"}

Returns a **ControlFormat** object that contains Microsoft Excel control properties. Read-only.

ControlFormat Property Example

This example removes the selected item from a list box. If `Shapes(2)` doesn't represent a list box, this example fails.

```
Set lbcf = Worksheets(1).Shapes(2).ControlFormat
lbcf.RemoveItem lbcf.ListIndex
```

DropDownLines Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDropDownLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDropDownLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDropDownLinesA"}

Returns or sets the number of list lines displayed in the drop-down portion of a combo box. Read/write **Long**.

Remarks

This property is ignored on the Apple Macintosh.

DropDownLines Property Example

This example creates a combo box with 10 list lines.

```
With Worksheets(1).Shapes.AddFormControl(xlDropDown, _
      Left:=10, Top:=10, Width:=100, Height:=10)
   .ControlFormat.DropDownLines = 10
End With
```

FormControlType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFormControlTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFormControlTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFormControlTypeA"}

Returns the Microsoft Excel control type. Can be one of the following **XlFormControl** constants: **xlButtonControl**, **xlCheckBox**, **xlDropDown**, **xlEditBox**, **xlGroupBox**, **xlLabel**, **xlListBox**, **xlOptionButton**, **xlScrollBar**, or **xlSpinner**. Read-only **Long**.

Remarks

You cannot use this property with ActiveX controls (the **Type** property for the **Shape** object must return **msoFormControl**).

FormControlType Property Example

This example clears all the Microsoft Excel check boxes on worksheet one.

```
For Each s In Worksheets(1).Shapes
   If s.Type = msoFormControl Then
      If s.FormControlType = xlCheckBox Then _
         s.ControlFormat.Value = False
   End If
Next
```

LargeChange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLargeChangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLargeChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLargeChangeA"}

Returns or sets the amount that the scroll box increments or decrements for a page scroll (when the user clicks in the scroll bar body region). Read/write **Long**

LargeChange Property Example

This example creates a scroll bar and sets its linked cell, minimum, maximum, large change, and small change values.

```
Set sb = Worksheets(1).Shapes.AddFormControl(xlScrollBar, _
   Left:=10, Top:=10, Width:=10, Height:=200)
With sb.ControlFormat
   .LinkedCell = "D1"
   .Max = 100
   .Min = 0
   .LargeChange = 10
   .SmallChange = 2
End With
```

LinkFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLinkFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLinkFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLinkFormatA"}

Returns a **LinkFormat** object that contains linked OLE object properties. Read-only.

LinkFormat Property Example

This example updates all linked OLE objects on worksheet one.

```
For Each s In Worksheets(1).Shapes
   If s.Type = msoLinkedOLEObject Then s.LinkFormat.Update
Next
```

ListCount Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproListCountC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproListCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproListCountA"}

Returns the number of entries in a list box or combo box. Returns 0 (zero) if there are no entries in the list. Read-only **Long**.

ListCount Property Example

This example adjusts a combo box to display all entries in its list. If `Shapes(1)` does not represent a combo box, this example fails.

```
Set cf = Worksheets(1).Shapes(1).ControlFormat
cf.DropDownLines = cf.ListCount
```

ListIndex Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproListIndexC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproListIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproListIndexA"}

Returns or sets the index number of the currently selected item in a list box or combo box. Read/write **Long**.

Remarks

You cannot use this property ith multiselect list boxes.

ListIndex Property Example

This example removes the selected item from a list box. If `Shapes(2)` doesn't represent a list box, this example fails.

```
Set lbcf = Worksheets(1).Shapes(2).ControlFormat
lbcf.RemoveItem lbcf.ListIndex
```

LockedText Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLockedTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLockedTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLockedTextA"}

True if the text in the specified object will be locked to prevent changes when the workbook is protected. Read/write **Boolean**.

LockedText Property Example

This example locks text in embedded chart one when the workbook is protected.

```
Worksheets(1).ChartObjects(1).LockedText = True
```

Max Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMaxC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMaxX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMaxA"}

Returns or sets the maximum value of a scroll bar or spinner range. The scroll bar or spinner won't take on values greater than this maximum value. Read/write **Long**.

For information about using the **Max** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Remarks

The value of the **Max** property must be greater than the value of the **Min** property.

Max Property Example

This example creates a scroll bar and sets its linked cell, minimum, maximum, large change, and small change values.

```
Set sb = Worksheets(1).Shapes.AddFormControl(xlScrollBar, _
   Left:=10, Top:=10, Width:=10, Height:=200)
With sb.ControlFormat
   .LinkedCell = "D1"
   .Max = 100
   .Min = 0
   .LargeChange = 10
   .SmallChange = 2
End With
```

Min Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMinC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMinX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMinA"}

Returns or sets the minumum value of a scroll bar or spinner range. The scroll bar or spinner won't take on values less than this minimum value. Read/write **Long**.

For information about using the **Min** worksheet function in Visual Basic, see Using Worksheet Functions in Visual Basic.

Remarks

The value of the **Min** property must be less than the value of the **Max** property.

Min Property Example

This example creates a scroll bar and sets its linked cell, minimum, maximum, large change, and small change values.

```
Set sb = Worksheets(1).Shapes.AddFormControl(xlScrollBar, _
   Left:=10, Top:=10, Width:=10, Height:=200)
With sb.ControlFormat
   .LinkedCell = "D1"
   .Max = 100
   .Min = 0
   .LargeChange = 10
   .SmallChange = 2
End With
```

MultiSelect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMultiSelectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMultiSelectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMultiSelectA"}

Returns or sets the selection mode of the specified list box. Can be one of the following constants: **xlNone**, **xlSimple**, or **xlExtended**. Read/write **Long**.

Remarks

Single select (**xlNone**) allows only one item at a time to be selected. Clicking the mouse or pressing the SPACEBAR cancels the selection and selects the clicked item.

Simple multiselect (**xlSimple**) toggles the selection on an item in the list when click it with the mouse or press the SPACEBAR when the focus is on the item. This mode is appropriate for pick lists, in which there are often multiple items selected.

Extended multiselect (**xlExtended**) usually acts like a single-selection list box, so when you click an item, you cancel all other selections. When you hold down SHIFT while clicking the mouse or pressing an arrow key, you select items sequentially from the current item. When you hold down CTRL while clicking the mouse, you add single items to the list. This mode is appropriate when multiple items are allowed but not often used.

You can use the **Value** or **ListIndex** property to return and set the selected item in a single-select list box.

You cannot link multiselect list boxes by using the **LinkedCell** property.

MultiSelect Property Example

This example creates a simple multiselect list box.

```
Set lb = Worksheets(1).Shapes.AddFormControl(xlListBox, _
   Left:=10, Top:=10, Height:=100, Width:100)
lb.ControlFormat.MultiSelect = xlSimple
```

OLEFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOLEFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOLEFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOLEFormatA"}

Returns an **OLEFormat** object that contains OLE object properties. Read-only.

OLEFormat Property Example

This example activates an OLE object. If `Shapes`(1) doesn't represent an embedded OLE object, this example fails..

```
Worksheets(1).Shapes(1).OLEFormat.Activate
```

OnAction Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOnActionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOnActionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOnActionA"}

Returns or sets the name of a macro that's run when the specified object is clicked. Read/write **String**.

Remarks

Setting this property for a menu item overrides any custom help information set up for the menu item with the information set up for the assigned macro.

OnAction Property Example

This example causes Microsoft Excel to run the ShapeClick procedure whenever shape one is clicked.

```
Worksheets(1).Shapes(1).OnAction = "ShapeClick"
```

OnWindow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOnWindowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOnWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOnWindowA"}

Returns or sets the name of the procedure that's run whenever you activate a window. Read/write **String**.

Remarks

The procedure specified by this property isn't run when other procedures switch to the window or when a command to switch to a window is received through a DDE channel. Instead, the procedure responds to the user's actions, such as clicking a window with the mouse, clicking **Go To** on the **Edit** menu, and so on.

If a worksheet or macro sheet has an Auto_Activate or Auto_Deactivate macro defined for it, those macros will be run after the procedure specified by the **OnWindow** property.

OnWindow Property Example

This example causes the WindowActivate procedure to be run whenever window one is activated.

```
ThisWorkbook.Windows(1).OnWindow = "WindowActivate"
```

SmallChange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSmallChangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSmallChangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSmallChangeA"}

Returns or sets the amount that the scroll bar or spinner is incremented or decremented for a line scroll (when the user clicks an arrow). Read/write **Long**.

SmallChange Property Example

This example creates a scroll bar and sets its linked cell, minimum, maximum, large change, and small change values.

```
Set sb = Worksheets(1).Shapes.AddFormControl(xlScrollBar, _
   Left:=10, Top:=10, Width:=10, Height:=200)
With sb.ControlFormat
   .LinkedCell = "D1"
   .Max = 100
   .Min = 0
   .LargeChange = 10
   .SmallChange = 2
End With
```

Item Property (AddIns Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemAddInsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemAddInsObjA"}

Returns a single **AddIn** object from an **AddIns** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the add-in.

Item Property (AddIns Collection) Example

This example displays the status of the Analysis ToolPak add-in. Note that the string used as the index to the **AddIns** method is the **Title** property of the **AddIn** object.

```
If AddIns.Item("Analysis ToolPak").Installed = True Then
   MsgBox "Analysis ToolPak add-in is installed"
Else
   MsgBox "Analysis ToolPak add-in is not installed"
End If
```

Item Property (Areas Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemAreasObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemAreasObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemAreasObjA"}

Returns a single **Range** object from an **Areas** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the range.

Item Property (Areas Collection) Example

This example clears the first area in the current selection if the selection contains more than one area.

```
If Selection.Areas.Count <> 1 Then
   Selection.Areas.Item(1).Clear
End If
```

Item Property (Charts Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemChartsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemChartsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemChartsObjA"}

Returns a single **Chart** object from a **Charts** collection.

Syntax

expression.**Item(*Index*)**

expression An expression that returns a **Charts** object.

Index Required **Variant**. The name or index number of the chart.

Item Property (Charts Collection) Example

This example sets the number of units that the trendline on Chart1 extends forward and backward. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
With Charts.Item("Chart1").SeriesCollection(1).Trendlines(1)
    .Forward = 5
    .Backward = .5
End With
```

Item Property (HPageBreaks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemHPageBreaksObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemHPageBreaksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemHPageBreaksObjA"}

Returns a single **HPageBreak** object from an **HPageBreaks** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the horizontal page break.

Item Property (HPageBreaks Collection) Example

This example changes the location of horizontal page break one.

```
Worksheets(1).HPageBreaks.Item(1).Location = .Range("e5")
```

Item Property (Hyperlinks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemHyperlinksObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemHyperlinksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemHyperlinksObjA"}

Returns a single **Hyperlink** object from a **Hyperlinks** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the hyperlink.

Item Property (Hyperlinks Collection) Example

The folllowing example activates hyperlink one on cell E5.

```
Worksheets(1).Range("E5").Hyperlinks.Item(1).Follow
```

Item Property (Panes Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemPanesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemPanesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemPanesObjA"}

Returns a single **Pane** object from a **Panes** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the pane.

Item Property (Panes Collection) Example

This example splits the window in which worksheet one is displayed and then scrolls through the pane in the lower-left corner of the window until row five is at the top of the pane.

```
Worksheets(1).Activate
ActiveWindow.Split = True
ActiveWindow.Panes.Item(3).ScrollRow = 5
```

Item Property (RecentFiles Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemRecentFilesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemRecentFilesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemRecentFilesObjA"}

Returns a single **RecentFile** object from a **RecentFiles** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the file.

Item Property (RecentFiles Collection) Example

This example opens file two in the list of recently used files.

```
Application.RecentFiles.Item(2).Open
```

Item Property (Sheets Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemSheetsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemSheetsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemSheetsObjA"}

Returns a single **Chart** or **Worksheet** object from a **Sheets** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**.The name or index number of the sheet.

Item Property (Sheets Collection) Example

This example activates Sheet1.

```
Sheets.Item("sheet1").Activate
```

Item Property (Styles Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemStylesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemStylesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemStylesObjA"}

Returns a single **Style** object from a **Styles** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the style.

Item Property (Styles Collection) Example

This example changes the Normal style for the active workbook by setting the style's **Bold** property.

```
ActiveWorkbook.Styles.Item("Normal").Font.Bold = True
```

Item Property (VPageBreaks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemVPageBreaksObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemVPageBreaksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemVPageBreaksObjA"}

Returns a single **VPageBreak** object from a **VPageBreaks** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the vertical page break.

Item Property (VPageBreaks Collection) Example

This example changes the location of vertical page break one.

```
Worksheets(1).VPageBreaks.Item(1).Location = .Range("e5")
```

Item Property (Windows Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemWindowsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemWindowsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemWindowsObjA"}

Returns a single **Window** object from a **Windows** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the window.

Item Property (Windows Collection) Example

This example maximizes the active window.

```
Windows.Item(1).WindowState = xlMaximized
```

Item Property (Workbooks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemWorkbooksObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemWorkbooksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemWorkbooksObjA"}

Returns a single **Workbook** object from a **Workbooks** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the workbook.

Item Property (Workbooks Collection) Example

This example sets the `wb` variable to the workbook for Myaddin.xla.

```
Set wb = Workbooks.Item("myaddin.xla")
```

Item Property (Worksheets Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemWorksheetsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemWorksheetsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemWorksheetsObjA"}

Returns a single **Worksheet** object from a **Worksheets** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the worksheet.

Item Property (Worksheets Collection) Example

This example changes the location of vertical page break one.

```
Worksheets.Item(1).VPageBreaks(1).Location = .Range("e5")
```

Item Method (Axes Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemAxesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemAxesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemAxesObjA"}

Returns a single **Axis** object from an **Axes** collection.

Syntax

expression.**Item(*Type, AxisGroup*)**

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

Type Required **Variant**. The axis type. Can be one of the following **XlAxisType** constants: **xlValue**, **xlCategory**, or **xlSeriesAxis** (**xlSeriesAxis** is valid only for 3-D charts).

AxisGroup Optional **Variant**. The axis group. Can be one of the following **XlAxisGroup** constants: **xlPrimary** or **xlSecondary**. The default value is **xlPrimary**.

Item Method (Axes Collection) Example

This example sets the title text for the category axis on Chart1.

```
With Charts("chart1").Axes.Item(xlCategory)
    .HasTitle = True
    .AxisTitle.Caption = "1994"
End With
```

Item Method (CalculatedFields Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemCalculatedFieldsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemCalculatedFieldsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemCalculatedFieldsObjA"}

Returns a single **PivotField** object from a **CalculatedFields** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the pivot field.

Item Method (CalculatedFields Collection) Example

This example sets the formula for calculated field one.

```
Worksheets(1).PivotTables(1).CalculatedFields.Item(1) _
    .Formula = "=Revenue - Cost"
```

Item Method (CalculatedItems Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemCalculatedItemsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemCalculatedItemsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemCalculatedItemsObjA"}

Returns a single **PivotItem** object from a **CalculatedItems** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the pivot item.

Item Method (CalculatedItems Collection) Example

This example hides calculated item one.

```
Worksheets(1).PivotTables(1).PivotFields("year") _
    .CalculatedItems.Item(1).Visible = False
```

Item Method (ChartGroups Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemChartGroupsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemChartGroupsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemChartGroupsObjA"}

Returns a single **ChartGroup** object from a **ChartGroups** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the chart group.

Item Method (ChartGroups Collection) Example

This example adds drop lines to chart group one on chart sheet one.

```
Charts(1).ChartGroups.Item(1).HasDropLines = True
```

Item Method (ChartObjects Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemChartObjectsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemChartObjectsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemChartObjectsObjA"}

Returns a single **ChartObject** object from a **ChartObjects** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the embedded chart.

Item Method (ChartObjects Collection) Example

This example activates embedded chart one.

```
Worksheets("sheet1").ChartObjects.Item(1).Activate
```

Item Method (Comments Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemCommentsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemCommentsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemCommentsObjA"}

Returns a single **Comment** object from a **Comments** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the comment.

Item Method (Comments Collection) Example

This example example hides comment two.

```
Worksheets(1).Comments.Item(2).Visible = False
```

Item Method (CustomViews Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemCustomViewsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemCustomViewsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemCustomViewsObjA"}

Returns a single **CustomView** object from a **CustomViews** collection.

Syntax

expression.**Item(*ViewName*)**

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

ViewName Required **Variant**. The name or index number of the custom view.

Item Method (CustomViews Collection) Example

This example includes print settings in the custom view named "Current Inventory."

```
ThisWorkbook.CustomViews.Item("Current Inventory").PrintSettings = True
```

Item Method (DataLabels Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemDataLabelsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemDataLabelsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemDataLabelsObjA"}

Returns a single **DataLabel** object from a **DataLabels** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the data label.

Item Method (DataLabels Collection) Example

This example sets the number format for the fifth data label in series one in embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Chart _
    .SeriesCollection(1).DataLabels.Item(5).NumberFormat = "0.000"
```

Item Method (FormatConditions Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemFormatConditionsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemFormatConditionsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemFormatConditionsObjA"}

Returns a single **FormatCondition** object from a **FormatConditions** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the conditional format.

Item Method (FormatConditions Collection) Example

This example sets format properties for an existing conditional format for cells E1:E10.

```
With Worksheets(1).Range("e1:e10").FormatConditions.Item(1)
    With .Borders
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = 6
    End With
End With
```

Item Method (LegendEntries Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemLegendEntriesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemLegendEntriesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemLegendEntriesObjA"}

Returns a single **LegendEntry** object from a **LegendEntries** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the legend entry.

Item Method (LegendEntries Collection) Example

This example changes the font for the text of the legend entry at the top of the legend (this is usually the legend for series one) in embedded chart one on Sheet1.

```
Worksheets("sheet1").ChartObjects(1).Chart _
    .Legend.LegendEntries.Item(1).Font.Italic = True
```

Item Method (Names Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemNamesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemNamesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemNamesObjA"}

Returns a single **Name** object from a **Names** collection.

Syntax

expression.**Item(*Index, IndexLocal, RefersTo*)**

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

Index Optional **Variant**. The name or number of the defined name to be returned.

IndexLocal Optional **Variant**. The name of the defined name, in the language of the user. No names will be translated if you use this argument.

RefersTo Optional **Variant**. What the name refers to. You use this argument to identify a name by what it refers to.

Remarks

You must specify one, and only one, of these three arguments.

Item Method (Names Collection) Example

This example deletes the name "mySortRange" from the active workbook.

```
ActiveWorkbook.Names.Item("mySortRange").Delete
```

Item Method (ODBCErrors Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemODBCErrorsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemODBCErrorsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemODBCErrorsObjA"}

Returns a single **ODBCError** object from an **ODBCErrors** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the **ODBCError** object.

Item Method (ODBCErrors Collection) Example

This example displays an ODBC error.

```
Set er = Application.ODBCErrors.Item(1)
MsgBox "The following error occurred:" &
   er.ErrorString & " : " & er.SqlState
```

Item Method (OLEObjects Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemOLEObjectsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemOLEObjectsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemOLEObjectsObjA"}

Returns a single **OLEObject** object from an **OLEObjects** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the **OLEObject** object.

Item Method (OLEObjects Collection) Example

This example deletes OLE object one from Sheet1.

```
Worksheets("sheet1").OLEObjects.Item(1).Delete
```

Item Method (Parameters Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemParametersObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemParametersObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemParametersObjA"}

Returns a single **Parameter** object from a **Parameters** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the parameter.

Item Method (Parameters Collection) Example

This example modifies the parameter prompt string.

```
With Worksheets(1).QueryTables(1).Parameters.Item(1)
    .SetParam xlPrompt, "Please " & .PromptString
End With
```

Item Method (PivotCaches Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemPivotCachesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemPivotCachesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemPivotCachesObjA"}

Returns a single **PivotCache** object from a **PivotCaches** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the pivot cache.

Item Method (PivotCaches Collection) Example

This example refreshes cache one.

```
ActiveWorkbook.PivotCaches.Item(1).Refresh
```

Item Method (PivotFields Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemPivotFieldsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemPivotFieldsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemPivotFieldsObjA"}

Returns a single **PivotField** object from a **PivotFields** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the pivot field.

Item Method (PivotFields Collection) Example

This example makes the Year field a row field in PivotTable one on Sheet3.

```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields.Item("year").Orientation = xlRowField
```

Item Method (PivotFormulas Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemPivotFormulasObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemPivotFormulasObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemPivotFormulasObjA"}

Returns a single **PivotFormula** object from a **PivotFormulas** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the pivot formula.

Item Method (PivotFormulas Collection) Example

This example displays the formula for pivot formula one.

```
MsgBox Worksheets(1).PivotTables(1).PivotFormulas.Item(1).Formula
```

Item Method (PivotItems Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemPivotItemsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemPivotItemsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemPivotItemsObjA"}

Returns a single **PivotItem** object from a **PivotItems** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the pivot item.

Item Method (PivotItems Collection) Example

This example hides the "1998" item in PivotTable one on Sheet3.

```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields("year").PivotItems.Item("1998").Visible = False
```

Item Method (PivotTables Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemPivotTablesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemPivotTablesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemPivotTablesObjA"}

Returns a single **PivotTable** object from a **PivotTables** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the PivotTable.

Item Method (PivotTables Collection) Example

This example makes the Year field a row field in PivotTable one on Sheet3.

```
Worksheets("sheet3").PivotTables.Item(1) _
    .PivotFields("year").Orientation = xlRowField
```

Item Method (Points Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemPointsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemPointsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemPointsObjA"}

Returns a single **Point** object from a **Points** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Long**. The index number of the point.

Item Method (Points Collection) Example

This example sets the marker style for the third point in series one in embedded chart one on worksheet one. The specified series must be a 2-D line, scatter, or radar series.

```
Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection(1).Points.Item(3).MarkerStyle = xlDiamond
```

Item Method (QueryTables Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemQueryTablesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemQueryTablesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemQueryTablesObjA"}

Returns a single **QueryTable** object from a **QueryTables** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The index number of the query table.

Item Method (QueryTables Collection) Example

This example sets a query table so that formulas to the right of the query table are automatically updated whenever it's refreshed.

```
Sheets("sheet1").QueryTables.Item(1).FillAdjacentFormulas = True
```

Item Method (Scenarios Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemScenariosObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemScenariosObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemScenariosObjA"}

Returns a single **Scenario** object from a **Scenarios** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the scenario.

Item Method (Scenarios Collection) Example

This example shows the scenario named "Typical" on the worksheet named "Options."

```
Worksheets("options").Scenarios.Item("typical").Show
```

Item Method (SeriesCollection Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemSeriesCollectionObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemSeriesCollectionObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemSeriesCollectionObjA"}

Returns a single **Series** object from a **SeriesCollection** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the series.

Item Method (SeriesCollection Collection) Example

This example sets the number of units that the trendline on Chart1 extends forward and backward. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
With Charts("Chart1").SeriesCollection.Item(1).Trendlines.Item(1)
    .Forward = 5
    .Backward = .5
End With
```

Item Method (ShapeRange Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemShapeRangeObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemShapeRangeObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemShapeRangeObjA"}

Returns a single **Shape** object from a **ShapeRange** object.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the shape.

Item Method (ShapeRange Collection) Example

This example sets the **OnAction** property for shape two in a shape range. If the `sr` variable doesn't represent a **ShapeRange** object, this example fails.

```
sr.Item(2).OnAction = "ShapeAction"
```

Item Method (Shapes Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemShapesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemShapesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemShapesObjA"}

Returns a single **Shape** object from a **Shapes** collection.

Syntax

expression.**Item(*Index*)**

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

Index Required **Variant**. The name or index number of the shape.

Item Method (Shapes Collection) Example

This example sets the **OnAction** property for shape two in a **Shapes** collection. If the `ss` variable doesn't represent a **Shapes** object, this example fails.

```
ss.Item(2).OnAction = "ShapeAction"
```

Item Method (Trendlines Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemTrendlinesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemTrendlinesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemTrendlinesObjA"}

Returns a single **Trendline** object from a **Trendlines** collection.

Syntax

expression.**Item(*Index*)**

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

Index Optional **Variant**. The name or index number of the trendline.

Item Method (Trendlines Collection) Example

This example sets the number of units that the trendline on Chart1 extends forward and backward. The example should be run on a 2-D column chart that contains a single series with a trendline.

```
With Charts("Chart1").SeriesCollection(1).Trendlines.Item(1)
    .Forward = 5
    .Backward = .5
End With
```

Microsoft Excel controls

Controls you add with the **Forms** toolbar. Don't confuse Microsoft Exel controls with ActiveX controls (controls you add with the **Control Toolbox**).

ActiveX controls

Controls you add with the **Control Toolbox**. ActiveX controls are embedded OLE objects. Don't confuse ActiveX controls with Microsoft Excel controls (controls you add with the **Forms** toolbar).

Add Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddC"}

Adds an object to a collection. For more information, click a collection in the following list.

AddIns
CalculatedFields
CalculatedItems
ChartObjects
Charts
CustomViews
FormatConditions
HPageBreaks
Hyperlinks
Names
OLEObjects
Parameters
PivotFormulas
PivotItems
QueryTables
RecentFiles
Scenarios
SeriesCollection
Sheets
Styles
Trendlines
Validation
VPageBreaks
Workbooks
Worksheets

Group Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGroupC"}

For more information, click one of the following objects.

Range

ShapeRange

Ungroup Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUngroupC"}

For more information, click one of the following objects.

Range

Shape

ShapeRange

Range Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRangeC"}

For more information, click one of the following objects.

Application

Hyperlink

Range

Shapes

Worksheet

Modify Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthModifyC"}

Modifies the data validation or conditional format. For more information, click the object you want to modify.

FormatCondition

Validation

BarShape Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBarShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBarShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBarShapeA"}

Returns or sets the shape used with the 3-D bar or column chart. Can be one of the following **XlBarShape** constants: **xlBox**, **xlConeToMax**, **xlConeToPoint**, **xlCylinder**, **xlPyramidToMax**, or **xlPyramidToPoint**. Read/write **Long**.

BarShape Property Example

This example sets the shape used with series one on chart one.

```
Charts(1).SeriesCollection(1).BarShape = xlConeToPoint
```

CharacterType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCharacterTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCharacterTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCharacterTypeA"}

Not used in U.S. English Microsoft Excel.

PivotSelect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPivotSelectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPivotSelectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPivotSelectA"}

Selects part of a PivotTable.

Syntax

expression.**PivotSelect(*Name, Mode*)**

expression An expression that returns a **PivotTable** object.

Name Required **String**. The selection, in standard PivotTable selection format.

Mode Required Long. Specifies the structured selection mode. Can be one of the following **XlPTSelectionMode** constants: **xlBlanks**, **xlButton**, **xlDataAndLabel**, **xlDataOnly**, **xlLabelOnly**, or **xlOrigin**.

PivotSelect Method Example

This example selects all date labels in PivotTable one.

```
Worksheets(1).PivotTables(1).PivotSelect "date[All]", xlLabelOnly
```

PivotSelection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPivotSelectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPivotSelectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPivotSelectionA"}

Returns or sets the PivotTable selection, in standard PivotTable selection format. Read/write **String**.

Remarks

Setting this property is equivalent to calling the **PivotSelect** method with the ***Mode*** argument set to **xlDataAndLabel**.

PivotSelection Property Example

This example selects the data and label for the salesperson named "Bob" in PivotTable one.

```
Worksheets(1).PivotTables(1).PivotSelection = "Salesman[Bob]"
```

PurgeChangeHistoryNow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPurgeChangeHistoryNowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPurgeChangeHistoryNowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPurgeChangeHistoryNowA"}

Removes entries from the change log for the specified workbook.

Syntax

expression.**PurgeChangeHistoryNow(*Days, SharingPassword*)**

expression An expression that returns a **Workbook** object.

Days Required **Long**. The number of days that changes in the change log are to be retained.

SharingPassword Optional **Variant**. The password that unprotects the workbook for sharing. If the workbook is protected for sharing with a password and this argument is omitted, the user is prompted for the password.

PurgeChangeHistoryNow Method Example

This example removes all changes that are more than one day old from the change log for the active workbook.

```
ActiveWorkbook.PurgeChangeHistoryNow Days:=1
```

Range Property (Hyperlink Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRangeHyperlinkObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRangeHyperlinkObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRangeHyperlinkObjA"}

Returns a **Range** object that represents the range the specified hyperlink is attached to. Read-only.

Range Property (Hyperlink Object) Example

This example scrolls through the workbook window until the hyperlink range is in the upper-left corner of the active window.

```
Workbooks(1).Activate
Set hr = ActiveSheet.Hyperlinks(1).Range
ActiveWindow.ScrollRow = hr.Row
ActiveWindow.ScrollColumn = hr.Column
```

Shapes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShapesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShapesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShapesA"}

Returns a **Shapes** object that represents all the shapes on the worksheet or chart sheet. 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 blue dashed line to worksheet one.

```
With Worksheets(1).Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```

Shape Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShapeA"}

Returns a **Shape** object that represents the shape attached to the specified comment or hyperlink.

Shape Property Example

This example selects comment two on the active sheet.

```
ActiveSheet.Comments(2).Shape.Select
```

ShowError Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowErrorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowErrorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowErrorA"}

True if the data validation error message will be displayed whenever the user enters invalid data. Read/write **Boolean**.

ShowError Property Example

This example adds data validation to cell A10 on worksheet one. The input value must be from 5 through 10; if the user types invalid data, an error message is displayed but no input message is displayed.

```
With Worksheets(1).Range("A10").Validation
    .Add Type:=xlValidateWholeNumber, AlertStyle:=xlValidAlertStop, _
        Operator:=xlBetween, Formula1:="5", Formula2:="10"
    .ErrorMessage = "value must be between 5 and 10"
    .ShowInput = False
    .ShowError = True
End With
```

ShowInput Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproShowInputC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproShowInputX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproShowInputA"}

True if the data validation input message will be displayed whenever the user selects a cell in the data validation range. Read/write **Boolean**.

ShowInput Property Example

This example adds data validation to cell A10. The input value must be from 5 through 10; if the user types invalid data, an error message is displayed but no input message is displayed.

```
With Worksheets(1).Range("A10").Validation
    .Add Type:=xlValidateWholeNumber, AlertStyle:=xlValidAlertStop, _
        Operator:=xlBetween, Formula1:="5", Formula2:="10"
    .ErrorMessage = "value must be between 5 and 10"
    .ShowInput = False
    .ShowError = True
End With
```

Add Method (CalculatedFields or CalculatedItems Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddCalculatedFieldsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddCalculatedFieldsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddCalculatedFieldsObjA"}

Creates a new calculated field or calculated item. Returns a **PivotField** or **PivotItem** object.

Syntax

object.**Add(*Name, Formula*)**

object Required. An expression that returns a **CalculatedFields** or **CalculatedItems** object.

Name Required **String**. The name of the field or item.

Formula Required **String**. The formula for the field or item.

Add Method (CalculatedFields or CalculatedItems Collection) Example

This example adds a calculated field to PivotTable one.

```
Worksheets(1).PivotTables(1).CalculatedFields.Add "PxS", _
   "= Product * Sales"
```

Add Method (ChartObjects Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddChartObjectsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddChartObjectsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddChartObjectsObjA"}

Creates a new embedded chart. Returns a **ChartObject** object.

Syntax

object.**Add(*Left, Top, Width, Height*)**

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

Left, ***Top*** Required **Long**. The initial coordinates of the new object (in points), relative to the upper-left corner of cell A1 on a worksheet or to the upper-left corner of a chart.

Width, ***Height*** Required **Long**. The initial size of the new object, in points.

Add Method (ChartObjects Collection) Example

This example creates a new embedded chart.

```
Set co = Sheets("Sheet1").ChartObjects.Add(50, 40, 200, 100)
co.Chart..ChartWizard Source:=Worksheets("Sheet1").Range("A1:B2"), _
   Gallery:=xlColumn, Format:=6, PlotBy:=xlColumns, _
   CategoryLabels:=1, SeriesLabels:=0, HasLegend:=1
```

Add Method (HPageBreaks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddHPageBreaksObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddHPageBreaksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddHPageBreaksObjA"}

Adds a horizontal page break. Returns an **HPageBreak** object.

Syntax

object.**Add(*Before*)**

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

Before Required **Object**. A **Range** object. The range above which the new page break will be added.

Add Method (HPageBreaks Collection) Example

This example adds a horizontal page break above cell F25 and adds a vertical page break to the left of this cell.

```
With Worksheets(1)
    .HPageBreaks.Add .Range("F25")
    .VPageBreaks.Add .Range("F25")
End With
```

Add Method (Hyperlinks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddHyperlinksObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddHyperlinksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddHyperlinksObjA"}

Adds a hyperink to the specified range or shape. Returns a **Hyperlink** object.

Syntax

object.**Add(*Anchor, Address, SubAddress*)**

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

Anchor Required **Object**. The anchor for the hyperlink. Can be either a **Range** or **Shape** object.

Address Required **String**. The address of the hyperlink.

SubAddress Optional **Variant**. The subaddress of the hyperlink.

Add Method (Hyperlinks Collection) Example

This example adds a hyperlink to cell A5.

```
With Worksheets(1)
    .Hyperlinks.Add Anchor:=.Range("a5"), _
       Address:="http://www.microsoft.com"
End With
```

Add Method (PivotFormulas Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddPivotFormulasObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddPivotFormulasObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddPivotFormulasObjA"}

Creates a new pivot formula. Returns a **PivotFormula** object.

Syntax

object.**Add(*Formula*)**

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

Formula Required **String**. The new pivot formula.

Add Method (PivotFormulas Collection) Example

This example creates a new pivot formula.

```
Worksheets(1).PivotTables(1).PivotFormulas _
   .Add "Year['1998'] Apples = (Year['1997'] Apples) * 2"
```

Add Method (PivotItems Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddPivotItemsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddPivotItemsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddPivotItemsObjA"}

Creates a new pivot item. Returns a **PivotItem** object.

Syntax

object.**Add(*Name*)**

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

Name Required **String**. The name of the new pivot item.

Add Method (PivotItems Collection) Example

This example creates a new pivot item.

```
Worksheets(1).PivotTables(1).PivotFields("Year").Add "1998"
```

Add Method (RecentFiles Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddRecentFilesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddRecentFilesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddRecentFilesObjA"}

Adds a file to the list of recently used files. Returns a **RecentFile** object.

Syntax

object.**Add(*Name*)**

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

Name Required **String**. The file name.

Add Method (RecentFiles Collection) Example

This example adds "Oscar.xls" to the list of recently used files.

```
Application.RecentFiles.Add Name:="oscar.xls"
```

Add Method (VPageBreaks Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddVPageBreaksObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddVPageBreaksObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddVPageBreaksObjA"}

Adds a vertical page break. Returns a **VPageBreak** object.

Syntax

object.**Add(*Before*)**

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

Before Required **Object**. A **Range** object. The range to the left of which the new page break will be added.

Add Method (VPageBreaks Collection) Example

This example adds a horizontal page break above cell F25 and adds a vertical page break to the left of this cell.

```
With Worksheets(1)
    .HPageBreaks.Add .Range("F25")
    .VPageBreaks.Add .Range("F25")
End With
```

Paste Method (Floor or Walls Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPasteFloorObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPasteFloorObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPasteFloorObjA"}

Paste a picture from the Clipboard on the floor or walls of the specified chart.

Syntax

object.**Paste**

object Required. An expression that returns a **Floor** or **Walls** object.

OLE Programmatic Identifiers (Microsoft Excel)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmscProgrammaticIdentifiersC;vafctCreateObject;vafctGetObject;OLE Programmatic Identifiers"}

You use an OLE programmatic identifier (sometimes called a ProgID) to create an Automation object. Use one of the following OLE programmatic identifiers to create a Microsoft Excel object.

Use this identifier	To create this object
Excel.Application	**Application**
Excel.Application.8	**Application**
Excel.Chart	**Workbook** (returned **Chart** in previous versions of Microsoft Excel)
Excel.Chart.8	**Workbook**
Excel.Sheet	**Workbook** (returned **Worksheet** in previous versions of Microsoft Excel)
Excel.Sheet.8	**Workbook**

Remarks

Using "Excel.Chart," "Excel.Sheet," or "Excel.Application" with no version number creates an object in the most recent version of Microsoft Excel available on the machine where the macro is running.

standard PivotTable selection format

A string expression used to specify part of a PivotTable. The easiest way to understand the required syntax is to turn on the macro recorder, select cells in the PivotTable, and then study the recorded code.

You can refer to a particular cell only if the PivotTable selection string contains the names of all the items used to identify individual cells in the selection. The number of items in that string should be equal to number of fields in the view for a cell in a normal data area. If the cell is used in calculating a subtotal or grand total, the number of items in the string would be fewer (including 0 (zero) for the intersection of the column and row grand totals).

The item names in the string can appear in any order. If an item name is ambiguous because it appears in another field as well, it must be qualified by "Field[Item]." If an item name contains symbols and spaces, or if it doesn't start with an alphabetic character, it should be enclosed in single quotation marks. If an item name contains embedded single quotation marks, each of these marks must be converted to two single quotation marks. Quotation marks aren't required for spaces if the name is unambiguous, and they're not required for unqualified names that begin with numbers.

AutoMargins Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoMarginsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoMarginsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoMarginsA"}

True if Microsoft Excel automatically calculates text frame margins. Read/write **Boolean**.

Remarks

When this property is **True**, the **MarginLeft**, **MarginRight**, **MarginTop**, and **MaginBottom** properties are ignored.

AutoMargins Property Example

This example causes Microsoft Excel to automatically calculate text frame margins for text in shape one.

```
Worksheets(1).Shapes(1).TextFrame.AutoMargins = True
```

Accent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAccentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAccentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAccentA"}

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 = Worksheets(1)
With myDocument.Shapes
    .AddShape msoShapeOval, 180, 200, 280, 130
    With .AddCallout(msoCalloutTwo, 420, 170, 170, 40)
        .TextFrame.Characters.Text = "My oval"
        With .Callout
            .Accent = True
            .Border = False
        End With
    End With
End With
```

AddCallout Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddCalloutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddCalloutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddCalloutA"}

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 = Worksheets(1)
myDocument.Shapes.AddCallout(msoCalloutTwo, 50, 50, 200, 100).Callout.Angle
= msoCalloutAngle30
```

AddConnector Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddConnectorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddConnectorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddConnectorA"}

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

AddCurve Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddCurveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddCurveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddCurveA"}

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 = Worksheets(1)
myDocument.Shapes.AddCurve pts
```

AddLabel Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddLabelC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddLabelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddLabelA"}

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 one of the following **MsoTextOrientation** constants: **msoTextOrientationDownward**, **msoTextOrientationHorizontal**, **msoTextOrientationHorizontalRotatedFarEast**, **msoTextOrientationMixed**, **msoTextOrientationUpward**, **msoTextOrientationVertical**, or **msoTextOrientationVerticalFarEast**. The Far East constants are not used in U.S./English Microsoft Excel.

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 = Worksheets(1)
myDocument.Shapes.AddLabel(msoTextOrientationVertical, 100, 100, 60, 150) _
    .TextFrame.Characters.Text = "Test Label"
```

AddLine Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddLineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddLineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddLineA"}

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 = Worksheets(1)
With myDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```

AddNodes Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddNodesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddNodesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddNodesA"}

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 four segments to `myDocument`.

```
Set myDocument = Worksheets(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
```

AddOLEObject Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddOLEObjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddOLEObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddOLEObjectA"}

Creates an OLE object. Returns a **Shape** object that represents the new OLE object.

Syntax

expression.**AddOLEObject**(***ClassType, FileName, Link, DisplayAsIcon, IconFileName, IconIndex, IconLabel, Left, Top, Width, Height,***)

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

ClassType Optional **Variant**. (you must specify either ***ClassType*** or ***FileName***). A string that contains the programmatic identifier for the object to be created. If ***ClassType*** is specified, ***FileName*** and ***Link*** are ignored. For more information about programmatic identifiers, see OLE Programmatic Identifiers.

FileName Optional **Variant**. 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 ***ClassType*** or ***FileName*** argument for the object, but not both.

Link Optional **Variant**. **True** to link the OLE object to the file from which it was created. **False** to make the OLE object an independent copy of the file. If you specified a value for ***ClassType***, this argument must be **False**. The default value is **False**.

DisplayAsIcon Optional **Variant**. **True** to display the OLE object as an icon. The default value is **False**.

IconFileName Optional **StriVariantng**. The file that contains the icon to be displayed.

IconIndex Optional **Variant**. 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 **Variant**. A label (caption) to be displayed beneath the icon.

Left, ***Top*** Optional **Variant**. The position (in points) of the upper-left corner of the new object relative to the upper-left corner of the document. The default value is 0 (zero).

Width, ***Height*** Optional **Variant**. The initial dimensions of the OLE object, in points.

AddOLEObject Method Example

This example adds a linked Word document to `myDocument`.

```
Set myDocument = Worksheets(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 command button to `myDocument`.

```
Set myDocument = Worksheets(1)
myDocument.Shapes.AddOLEObject Left:=100, Top:=100, Width:=100,
Height:=200, _
    ClassType:="Forms.CommandButton.1"
```

AddPicture Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddPictureC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddPictureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddPictureA"}

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 = Worksheets(1)
myDocument.Shapes.AddPicture "c:\microsoft office\clipart\music.bmp", _
    True, True, 100, 100, 70, 70
```

AddPolyline Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddPolylineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddPolylineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddPolylineA"}

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 = Worksheets(1)
myDocument.Shapes.AddPolyline triArray
```

AddShape Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddShapeA"}

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 = Worksheets(1)
myDocument.Shapes.AddShape msoShapeRectangle, 50, 50, 100, 200
```

AddTextbox Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddTextboxC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddTextboxX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddTextboxA"}

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 one of the following **MsoTextOrientation** constants: **msoTextOrientationDownward**, **msoTextOrientationHorizontal**, **msoTextOrientationHorizontalRotatedFarEast**, **msoTextOrientationMixed**, **msoTextOrientationUpward**, **msoTextOrientationVertical**, or **msoTextOrientationVerticalFarEast**. The Far East constants are not used in U.S. English Microsoft Excel.

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 = Worksheets(1)
myDocument.Shapes.AddTextbox(msoTextOrientationHorizontal, 100, 100, 200,
50) _
    .TextFrame.Characters.Text = "Test Box"
```

AddTextEffect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddTextEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddTextEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddTextEffectA"}

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 = Worksheets(1)
Set newWordArt =
myDocument.Shapes.AddTextEffect(PresetTextEffect:=msoTextEffect1,
Text:="Test", _
    FontName:="Arial Black", FontSize:=36, FontBold:=False,
FontItalic:=False, Left:=10, Top:=10)
```

Adjustments Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAdjustmentsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAdjustmentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAdjustmentsA"}

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 one on `myDocument`.

```
Set myDocument = Worksheets(1)
myDocument.Shapes(1).Adjustments(1) = 0.25
```

Align Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAlignC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAlignX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAlignA"}

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**. Not used in Microsoft Excel. Must be **False**.

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 = Worksheets(1)
myDocument.Shapes.SelectAll
Selection.ShapeRange.Align msoAlignLefts, False
```

Alignment Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAlignmentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAlignmentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAlignmentA"}

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 adds a WordArt object to worksheet one and then right aligns the WordArt.

```
Set mySh = Worksheets(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
```

Angle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAngleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAngleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAngleA"}

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 "callout1" on `myDocument`.

```
Set myDocument = Worksheets(1)
myDocument.Shapes("callout1").Callout.Angle = msoCalloutAngle90
```

Apply Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthApplyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthApplyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthApplyA"}

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 = Worksheets(1)
With myDocument
    .Shapes(1).PickUp
    .Shapes(2).Apply
End With
```

AutoAttach Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoAttachC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoAttachX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoAttachA"}

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

AutoLength Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoLengthA"}

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 = Worksheets(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":"xlmthAutomaticLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAutomaticLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAutomaticLengthA"}

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

AutoShapeType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoShapeTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoShapeTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoShapeTypeA"}

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 = Worksheets(1)
For Each s In myDocument.Shapes
    If s.AutoShapeType = msoShape16pointStar Then
        s.AutoShapeType = msoShape32pointStar
    End If
Next
```

AutoSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproAutoSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproAutoSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproAutoSizeA"}

True if the size of the specified object is changed automatically to fit text within its boundaries. Read/write **Boolean**.

AutoSize Property Example

This example adjusts the size of the text frame on shape one to fit its text.

```
Worksheets(1).Shapes(1).TextFrame.AutoSize = True
```

BeginArrowheadLength Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBeginArrowheadLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBeginArrowheadLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBeginArrowheadLengthA"}

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 = Worksheets(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":"xlproBeginArrowheadStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBeginArrowheadStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBeginArrowheadStyleA"}

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 = Worksheets(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":"xlproBeginArrowheadWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBeginArrowheadWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBeginArrowheadWidthA"}

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 = Worksheets(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

BeginConnect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthBeginConnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthBeginConnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthBeginConnectA"}

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

BeginConnected Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBeginConnectedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBeginConnectedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBeginConnectedA"}

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 = Worksheets(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":"xlproBeginConnectedShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBeginConnectedShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBeginConnectedShapeA"}

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 = Worksheets(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":"xlproBeginConnectionSiteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBeginConnectionSiteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBeginConnectionSiteA"}

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 = Worksheets(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":"xlmthBeginDisconnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthBeginDisconnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthBeginDisconnectA"}

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

BlackWhiteMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBlackWhiteModeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBlackWhiteModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBlackWhiteModeA"}

This property is not used in Microsoft Excel. It is provided only for compatibility with the drawing object models in other Microsoft Office applications.

Brightness Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproBrightnessC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproBrightnessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproBrightnessA"}

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 = Worksheets(1)
myDocument.Shapes(1).PictureFormat.Brightness = 0.3
```

BuildFreeform Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthBuildFreeformC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthBuildFreeformX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthBuildFreeformA"}

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 five vertices to `myDocument`.

```
Set myDocument = Worksheets(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
```

Callout Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCalloutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCalloutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCalloutA"}

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 = Worksheets(1)
With myDocument.Shapes
    .AddShape msoShapeOval, 180, 200, 280, 130
    With .AddCallout(msoCalloutTwo, 420, 170, 170, 40)
        .TextFrame.Characters.Text = "My oval"
        With .Callout
            .Accent = True
            .Border = False
        End With
    End With
End With
```

ColorType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproColorTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproColorTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproColorTypeA"}

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 = Worksheets(1)
myDocument.Shapes(1).PictureFormat.ColorType = msoPictureGrayScale
```

ConnectionSiteCount Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproConnectionSiteCountC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConnectionSiteCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConnectionSiteCountA"}

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 = Worksheets(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":"xlproConnectorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConnectorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConnectorA"}

True if the specified shape is a connector. Read-only **Long**.

Connector Property Example

This example deletes all connectors on `myDocument`.

```
Set myDocument = Worksheets(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":"xlproConnectorFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConnectorFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConnectorFormatA"}

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

Contrast Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproContrastC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproContrastX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproContrastA"}

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 = Worksheets(1)
myDocument.Shapes(1).PictureFormat.Contrast = 0.8
```

ConvertToShape Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthConvertToShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthConvertToShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthConvertToShapeA"}

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 = Worksheets(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
```

CropBottom Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproCropBottomC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCropBottomX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCropBottomA"}

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 = Worksheets(1)
myDocument.Shapes(3).PictureFormat.CropBottom = 20
```

Using this example, you can specify the percentage you want to crop 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":"xlproCropLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCropLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCropLeftA"}

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 = Worksheets(1)
myDocument.Shapes(3).PictureFormat.CropLeft = 20
```

Using this example, you can specify the percentage you want to crop 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":"xlproCropRightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCropRightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCropRightA"}

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 = Worksheets(1)
myDocument.Shapes(3).PictureFormat.CropRight = 20
```

Using this example, you can specify the percentage you want to cropp 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":"xlproCropTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproCropTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproCropTopA"}

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 = Worksheets(1)
myDocument.Shapes(3).PictureFormat.CropTop = 20
```

This example allows you to specify the percentage you want to crop 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
```

CustomDrop Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCustomDropC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCustomDropX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCustomDropA"}

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

CustomLength Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCustomLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCustomLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCustomLengthA"}

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

DashStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDashStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDashStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDashStyleA"}

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 = Worksheets(1)
With myDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```

Depth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDepthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDepthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDepthA"}

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 = Worksheets(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
```

Distribute Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthDistributeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthDistributeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthDistributeA"}

Horizontally or vertically distributes the shapes in the specified range of shapes.

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**. Not used in Microsoft Excel. Must be **False**.

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. The leftmost shape retains its position.

```
Set myDocument = Worksheets(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
```

Drop Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDropC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDropX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDropA"}

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 = Worksheets(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":"xlproDropTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDropTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDropTypeA"}

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 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 = Worksheets(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
```

EditingType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEditingTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEditingTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEditingTypeA"}

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 = Worksheets(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
```

EndArrowheadLength Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEndArrowheadLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndArrowheadLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndArrowheadLengthA"}

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 = Worksheets(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":"xlproEndArrowheadStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndArrowheadStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndArrowheadStyleA"}

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 = Worksheets(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":"xlproEndArrowheadWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndArrowheadWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndArrowheadWidthA"}

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 = Worksheets(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

EndConnect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthEndConnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthEndConnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthEndConnectA"}

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

EndConnected Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEndConnectedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndConnectedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndConnectedA"}

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 = Worksheets(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":"xlproEndConnectedShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndConnectedShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndConnectedShapeA"}

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

EndConnectionSite Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEndConnectionSiteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEndConnectionSiteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEndConnectionSiteA"}

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

EndDisconnect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthEndDisconnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthEndDisconnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthEndDisconnectA"}

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

ExtrusionColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproExtrusionColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproExtrusionColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproExtrusionColorA"}

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 = Worksheets(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":"xlproExtrusionColorTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproExtrusionColorTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproExtrusionColorTypeA"}

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 = Worksheets(1)
With myDocument.Shapes(1).ThreeD
    If .ExtrusionColorType = msoExtrusionColorAutomatic Then
        .ExtrusionColor.RGB = RGB(240, 235, 16)
    End If
End With
```

Fill Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFillC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFillX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFillA"}

Returns a **FillFormat** object that contains fill formatting properties for the specified chart or 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 = Worksheets(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":"xlmthFlipC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthFlipX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthFlipA"}

Flips the specified shape around its 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 = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeRightTriangle, 10, 10, 50,
50).Duplicate
    .Fill.ForeColor.RGB = RGB(255, 0, 0)
    .Flip msoFlipVertical
End With
```

FontBold Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFontBoldC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFontBoldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFontBoldA"}

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 = Worksheets(1)
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.FontBold = True
    End If
End With
```

FontItalic Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFontItalicC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFontItalicX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFontItalicA"}

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 = Worksheets(1)
myDocument.Shapes("WordArt 4").TextEffect.FontItalic = True
```

FontName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFontNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFontNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFontNameA"}

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 = Worksheets(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":"xlproFontSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFontSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFontSizeA"}

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 = Worksheets(1)
myDocument.Shapes("WordArt 4").TextEffect.FontSize = 16
```

Gap Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproGapC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGapX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGapA"}

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 = Worksheets(1)
myDocument.Shapes(1).Callout.Gap = 3
```

Group Method (ShapeRange Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthGroupShapeRangeObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthGroupShapeRangeObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthGroupShapeRangeObjA"}

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 (ShapeRange Object) 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 = Worksheets(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":"xlproGroupItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproGroupItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproGroupItemsA"}

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 = Worksheets(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":"xlproHorizontalFlipC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproHorizontalFlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproHorizontalFlipA"}

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 = Worksheets(1)
For Each s In myDocument.Shapes
    If s.HorizontalFlip Then s.Flip msoFlipHorizontal
    If s.VerticalFlip Then s.Flip msoFlipVertical
Next
```

IncrementBrightness Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementBrightnessC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementBrightnessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementBrightnessA"}

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 = Worksheets(1)
With myDocument.Shapes(1).Duplicate
    .PictureFormat.IncrementBrightness -0.2
    .IncrementLeft 50
    .IncrementTop 50
End With
```

IncrementContrast Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementContrastC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementContrastX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementContrastA"}

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 = Worksheets(1)
For Each s In myDocument.Shapes
    If s.Type = msoPicture Or s.Type = msoLinkedPicture Then
        s.PictureFormat.IncrementContrast 0.1
    End If
Next
```

IncrementLeft Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementLeftA"}

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 = Worksheets(1)
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```

IncrementOffsetX Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementOffsetXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementOffsetXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementOffsetXA"}

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 = Worksheets(1)
myDocument.Shapes(3).Shadow.IncrementOffsetX -3
```

IncrementOffsetY Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementOffsetYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementOffsetYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementOffsetYA"}

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 = Worksheets(1)
myDocument.Shapes(3).Shadow.IncrementOffsetY -3
```

IncrementRotation Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementRotationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementRotationMethodX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementRotationA"}

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 = Worksheets(1)
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```

IncrementRotationX Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementRotationXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementRotationXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementRotationXA"}

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 = Worksheets(1)
myDocument.Shapes(1).ThreeD.IncrementRotationX 10
```

IncrementRotationY Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementRotationYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementRotationYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementRotationYA"}

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 = Worksheets(1)
myDocument.Shapes(1).ThreeD.IncrementRotationY -10
```

IncrementTop Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthIncrementTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthIncrementTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthIncrementTopA"}

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 = Worksheets(1)
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```

KernedPairs Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproKernedPairsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproKernedPairsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproKernedPairsA"}

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 = Worksheets(1)
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.KernedPairs = True
    End If
End With
```

Length Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLengthA"}

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**.

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 "callout1" has a fixed length, this example specifies that the length of the first line segment in the callout named "callout2" will also be fixed at that length. For the example to work, both callouts must have multiple-segment lines.

```
Set myDocument = Worksheets(1)
With myDocument.Shapes
    With .Item("callout1").Callout
        If Not .AutoLength Then len1 = .Length
    End With
    If len1 Then .Item("callout2").Callout.CustomLength len1
End With
```

Line Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLineA"}

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 = Worksheets(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 = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeCross, 10, 10, 50, 70).Line
    .Weight = 8
    .ForeColor.RGB = RGB(255, 0, 0)
End With
```

LockAspectRatio Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproLockAspectRatioC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproLockAspectRatioX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproLockAspectRatioA"}

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 = Worksheets(1)
myDocument.Shapes.AddShape(msoShapeCube, 50, 50, 100, 200).LockAspectRatio
= True
```

MarginBottom Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarginBottomC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarginBottomX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarginBottomA"}

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 = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 250,
140).TextFrame
    .Characters.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```

MarginLeft Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarginLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarginLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarginLeftA"}

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 = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 250,
140).TextFrame
    .Characters.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```

MarginRight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarginRightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarginRightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarginRightA"}

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

```
Set myDocument = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 250,
140).TextFrame
    .Characters.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```

MarginTop Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproMarginTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproMarginTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproMarginTopA"}

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 = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 250,
140).TextFrame
    .Characters.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```

Nodes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNodesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNodesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNodesA"}

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 = Worksheets(1)
With myDocument.Shapes(3).Nodes
    .Insert 4, msoSegmentCurve, msoEditingSmooth, 210, 100
End With
```

NormalizedHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproNormalizedHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproNormalizedHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproNormalizedHeightA"}

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 = Worksheets(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
```

Obscured Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproObscuredC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproObscuredX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproObscuredA"}

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 = Worksheets(1)
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
    .Obscured = True
End With
```

OffsetX Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOffsetXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOffsetXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOffsetXA"}

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

OffsetY Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproOffsetYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproOffsetYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproOffsetYA"}

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

PickUp Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPickUpC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPickUpX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPickUpA"}

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 = Worksheets(1)
With myDocument
    .Shapes(1).PickUp
    .Shapes(2).Apply
End With
```

PictureFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPictureFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPictureFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPictureFormatA"}

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 = Worksheets(1)
With myDocument.Shapes(1).PictureFormat
    .Brightness = 0.3
    .Contrast = .75
End With
```

Points Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPointsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPointsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPointsA"}

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.

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 = Worksheets(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
```

PresetDrop Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthPresetDropC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthPresetDropX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthPresetDropA"}

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

PresetExtrusionDirection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPresetExtrusionDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetExtrusionDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetExtrusionDirectionA"}

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 = Worksheets(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":"xlproPresetLightingDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetLightingDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetLightingDirectionA"}

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 = Worksheets(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```

PresetLightingSoftness Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPresetLightingSoftnessC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetLightingSoftnessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetLightingSoftnessA"}

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 = Worksheets(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .PresetLightingSoftness = msoLightingBright
    .PresetLightingDirection = msoLightingLeft
End With
```

PresetMaterial Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPresetMaterialC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetMaterialX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetMaterialA"}

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 = Worksheets(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .PresetMaterial = msoMaterialWireFrame
End With
```

PresetShape Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPresetShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetShapeA"}

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 = Worksheets(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":"xlproPresetTextEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetTextEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetTextEffectA"}

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
msoTextEffect2
msoTextEffect3
msoTextEffect4
msoTextEffect5
msoTextEffect6
msoTextEffect7
msoTextEffect8
msoTextEffect9
msoTextEffect10
msoTextEffect11
msoTextEffect12
msoTextEffect13
msoTextEffect14
msoTextEffect15
msoTextEffect16
msoTextEffect17
msoTextEffect18
msoTextEffect19
msoTextEffect20
msoTextEffect21
msoTextEffect22
msoTextEffect23
msoTextEffect24
msoTextEffect25
msoTextEffect26
msoTextEffect27
msoTextEffect28
msoTextEffect29
msoTextEffect30
msoTextEffectMixed

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 = Worksheets(1)
For Each s In myDocument.Shapes
    If s.Type = msoTextEffect Then
        s.TextEffect.PresetTextEffect = msoTextEffect1
    End If
Next
```

PresetThreeDFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPresetThreeDFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPresetThreeDFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPresetThreeDFormatA"}

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 = Worksheets(1)
With myDocument.Shapes(1).ThreeD
    If .PresetThreeDFormat = msoPresetThreeDFormatMixed Then
        .SetThreeDFormat msoThreeD12
    End If
End With
```

Range Property (Shapes Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRangeShapesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRangeShapesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRangeShapesObjA"}

Returns a **ShapeRange** object that represents a subset of the shapes in a **Shapes** collection.

Syntax

expression.**Range**(***Index***)

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

Index Required **Variant**. The individual shapes to be included in the range. Can be an integer that specifies the index number of the shape, a string that specifies the name of the shape, or an array that contains either integers or strings.

Remarks

Although you can use the **Range** property to return any number of shapes, 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)`.

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"))
```

In Microsoft Excel, you cannot use this property to return a **ShapeRange** object containing all the **Shape** objects on a worksheet. Instead, use the following code:

```
Worksheets(1).Shapes.Select  ' select all shapes
set sr = Selection.ShapeRange' create ShapeRange
```

Range Property (Shapes Collection) Example

This example sets the fill pattern for shapes one and three on `myDocument`.

```
Set myDocument = Worksheets(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 = Worksheets(1)
Set myRange = myDocument.Shapes.Range(Array("Oval 4", "Rectangle 5"))
myRange.Fill.Patterned msoPatternHorizontalBrick
```

This example sets the fill pattern for shape one on `myDocument`.

```
Set myDocument = Worksheets(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 = Worksheets(1)
With myDocument.Shapes
    numShapes = .Count
    If numShapes > 1 Then
        numAutoShapes = 1
        ReDim autoShpArray(1 To numShapes)
        For i = 1 To numShapes
            If .Item(i).Type = msoAutoShape Then
                autoShpArray(numAutoShapes) = .Item(i).Name
                numAutoShapes = numAutoShapes + 1
            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
```

Regroup Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRegroupC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRegroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRegroupA"}

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
```

RerouteConnections Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthRerouteConnectionsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthRerouteConnectionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthRerouteConnectionsA"}

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 = Worksheets(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
```

ResetRotation Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthResetRotationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthResetRotationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthResetRotationA"}

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 = Worksheets(1)
myDocument.Shapes(1).ThreeD.ResetRotation
```

RotatedChars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRotatedCharsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRotatedCharsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRotatedCharsA"}

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 = Worksheets(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
```

RotationX Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRotationXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRotationXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRotationXA"}

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 = Worksheets(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":"xlproRotationYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRotationYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRotationYA"}

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 = Worksheets(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
```

ScaleHeight Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthScaleHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthScaleHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthScaleHeightA"}

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**. Specifies which 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 = Worksheets(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":"xlmthScaleWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthScaleWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthScaleWidthA"}

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**. Specifies which 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 = Worksheets(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
```

SegmentType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSegmentTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSegmentTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSegmentTypeA"}

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 = Worksheets(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
```

SelectAll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSelectAllC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSelectAllX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSelectAllA"}

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` and creates a **ShapeRange** object containing all the shapes.

```
Set myDocument = Worksheets(1)
myDocument.Shapes.SelectAll
Set sr = Selection.ShapeRange
```

SetEditingType Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetEditingTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetEditingTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetEditingTypeA"}

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 = Worksheets(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
```

SetExtrusionDirection Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetExtrusionDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetExtrusionDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetExtrusionDirectionA"}

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 = Worksheets(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```

SetPosition Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetPositionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetPositionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetPositionA"}

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 = Worksheets(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":"xlmthSetSegmentTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetSegmentTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetSegmentTypeA"}

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 = Worksheets(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
```

SetShapesDefaultProperties Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetShapesDefaultPropertiesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetShapesDefaultPropertiesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetShapesDefaultPropertiesA"}

Applies the formatting for the specified shape to the default shape. Shapes created after this method has been used will have this formatting applied by default.

Syntax

expression.**SetShapesDefaultProperties**

expression Required. An expression that returns a **Shape** or **ShapeRange** 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 = Worksheets(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
```

SetThreeDFormat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSetThreeDFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSetThreeDFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSetThreeDFormatA"}

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 = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeOval, 30, 30, 50, 25).ThreeD
    .Visible = True
    .SetThreeDFormat msoThreeD12
End With
```

Solid Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthSolidC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthSolidX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthSolidA"}

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

TextEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTextEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTextEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTextEffectA"}

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 = Worksheets(1)
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.FontBold = True
    End If
End With
```

ThreeD Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproThreeDC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproThreeDX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproThreeDA"}

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 = Worksheets(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
```

ToggleVerticalText Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthToggleVerticalTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthToggleVerticalTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthToggleVerticalTextA"}

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 = Worksheets(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":"xlproTrackingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTrackingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTrackingA"}

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 = Worksheets(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
```

TransparencyColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTransparencyColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransparencyColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransparencyColorA"}

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 = Worksheets(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":"xlproTransparentBackgroundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTransparentBackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTransparentBackgroundA"}

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 = Worksheets(1)
With myDocument.Shapes(1)
    With .PictureFormat
        .TransparentBackground = True
        .TransparencyColor = blueScreen
    End With
    .Fill.Visible = False
End With
```

Ungroup Method (Shape or ShapeRange Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthUngroupShapeObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthUngroupShapeObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthUngroupShapeObjA"}

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 (Shape or ShapeRange Object) Example

This example ungroups any grouped shapes and disassembles any pictures or OLE objects on `myDocument`.

```
Set myDocument = Worksheets(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 = Worksheets(1)
For Each s In myDocument.Shapes
    If s.Type = msoGroup Then s.Ungroup
Next
```

VerticalFlip Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproVerticalFlipC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVerticalFlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVerticalFlipA"}

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 = Worksheets(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":"xlproVerticesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproVerticesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproVerticesA"}

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 = Worksheets(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 = Worksheets(1)
With myDocument.Shapes
    .AddCurve .Item(1).Vertices
End With
```

ZOrder Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthZOrderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthZOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthZOrderA"}

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** or **ShapeRange** 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 used only in Microsoft Word.

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 = Worksheets(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":"xlproZOrderPositionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproZOrderPositionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproZOrderPositionA"}

Returns the position of the specified shape in 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 = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeOval, 100, 100, 100, 300)
    While .ZOrderPosition > 2
        .ZOrder msoSendBackward
    Wend
End With
```

Item Method (GroupShapes Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemGroupShapesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemGroupShapesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemGroupShapesObjA"}

Returns a single **Shape** object from a **GroupShapes** collection.

Syntax

expression.**Item**(***Index***)

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

Index Required **Variant**. The shape name or index number.

Item Method (ShapeNodes Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthItemShapeNodesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthItemShapeNodesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthItemShapeNodesObjA"}

Returns a single **ShapeNode** object from a **ShapeNodes** collection.

Syntax

expression.**Item**(***Index***)

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

Index Required **Variant**. The shape node index number.

Item Property (Adjustments Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproItemAdjustmentsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproItemAdjustmentsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproItemAdjustmentsObjA"}

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.

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, 100, 200, and 150, 250. 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 3, 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
cpArray(3, 1) = 150  ' The x-coordinate of the third point
cpArray(3, 2) = 250  ' The y-coordinate of the third point
```

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).

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.

default shape

An invisible shape that contains the initial properties for newly created shapes.

Add Method (QueryTables Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthAddQueryTablesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthAddQueryTablesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthAddQueryTablesObjA"}

Creates a new query table. Returns a **QueryTable** object that represents the new query table.

Syntax

expression.**Add(*Connection, Destination, Sql*)**

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

Connection Required **Variant**. The data source for the query table. Can be one of the following:

- A string containing an ODBC connection string.
- A **QueryTable** object from which the query information is initially copied, including the connection string and the SQL text, but not including the ***Destination*** range. Specifying a **QueryTable** object causes the ***Sql*** argument to be ignored.
- A DAO **RecordSet** object. Data is read from the DAO recordset. Microsoft Excel retains the DAO recordset until the query table is deleted or the connection is changed. The resulting query table cannot be edited.
- A Web query. A string in the form "URL;<url>", where "URL;" is required but not localized and the rest of the string is used for the URL of the Web query.
- Data Finder. A string in the form "FINDER;<data finder file path>" where "FINDER;" is required but not localized. The rest of the string is the path and file name of a Data Finder file (*.dqy or *.iqy). The file is read when the **Add** method is run; subsequent calls to the **Connection** property of the query table will return strings beginning with "ODBC;" or "URL;" as appropriate.

Destination Required **Range**. The cell in the upper-left corner of the query table destination range (the range where the resulting query table will be placed). The destination range must be on the worksheet that contains the **QueryTables** object specified by *expression*.

Sql Optional **Variant**. The SQL query string to be run on the ODBC data source. This argument is optional when you're using an ODBC data source (if you don't specify it here, you should set it by using the **Sql** property of the query table before the table is refreshed). This argument cannot be used when a **QueryTable** object or a DAO **Recordset** object is specified as the data source.

Remarks

A query created by this method isn't run until the **Refresh** method is called.

Add Method (QueryTables Collection) Example

This example creates a new query table.

```
sqlstring = "select 96Sales.totals from 96Sales where profit < 5"
connstring = _
   "ODBC;DSN=96SalesData;UID=Rep21;PWD=NUyHwYQI;Database=96Sales"
With ActiveSheet.QueryTables.Add(Connection:=connstring, _
      Destination:=Range("B1"), Sql:=sqlstring)
   .Refresh
End With
```

CancelRefresh Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlmthCancelRefreshC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlmthCancelRefreshX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlmthCancelRefreshA"}

Cancels all background queries for the specified query table. Use the **Refreshing** property to determine whether a background query is currently in progress.

Syntax

expression.**CancelRefresh**

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

CancelRefresh Method Example

This example cancels a query table refresh operation.

```
With Worksheets(1).QueryTables(1)
    If .Refreshing Then .CancelRefresh
End With
```

Connection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproConnectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproConnectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproConnectionA"}

Returns or sets a string that contains one of the following: ODBC settings that enable Microsoft Excel to connect to an ODBC data source; a URL that enables Microsoft Excel to connect to a Web data source; or a file that specifies a database or Web query. Read/write **String**.

Remarks

Setting the **Connection** property doesn't immediately initiate the connection to the data source. You must use the **Refresh** method to make the connection and retrieve the data.

For more information about the connection string syntax, see the **Add** method.

Connection Property Example

This example supplies new ODBC connection information for query table one.

```
Worksheets(1).QueryTables(1) _
    .Connection:="ODBC;DSN=96SalesData;UID=Rep21;PWD=NUyHwYQI;"
```

Destination Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproDestinationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproDestinationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproDestinationA"}

Returns the cell in the upper-left corner of the query table destination range (the range where the resulting query table will be placed). The destination range must be on the worksheet that contains the **QueryTable** object. Read-only **Range**.

Destination Property Example

This example scrolls through the active window until the upper-left corner of query table one is in the upper-left corner of the window.

```
Set d = Worksheets(1).QueryTables(1).Destination
With ActiveWindow
    .ScrollColumn = d.Column
    .ScrollRow = d.Row
End With
```

FetchedRowOverflow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFetchedRowOverflowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFetchedRowOverflowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFetchedRowOverflowA"}

True if the number of rows returned by the last use of the **Refresh** method is greater than the number of rows available on the worksheet. Read-only **Boolean**.

FetchedRowOverflow Property Example

This example refreshes query table one. If the number of rows returned by the query exceeds the number of rows available on the worksheet, an error message is displayed.

```
With Worksheets(1).QueryTables(1)
    .Refresh
    If .FetchedRowOverflow Then
        MsgBox "Query too large: please redefine."
    End If
End With
```

FieldNames Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFieldNamesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFieldNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFieldNamesA"}

True if field names from the data source appear as column headings for the returned data. The default value is **True**. Read/write **Boolean**.

FieldNames Property Example

This example sets query table one so that the field names don't appear in it.

```
Worksheets(1).QueryTables(1).FieldNames = False
```

FillAdjacentFormulas Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproFillAdjacentFormulasC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproFillAdjacentFormulasX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproFillAdjacentFormulasA"}

True if formulas to the right of the specified query table are automatically updated whenever the query table is refreshed. Read/write **Boolean**.

FillAdjacentFormulas Property Example

This example sets query table one so that formulas to the right of it are automatically updated whenever the query table is refreshed.

```
Sheets("sheet1").QueryTables(1).FillAdjacentFormulas = True
```

ODBCErrors Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproODBCErrorsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproODBCErrorsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproODBCErrorsA"}

Returns an **ODBCErrors** collection that contains all the ODBC errors generated by the most recent query table or PivotTable operation. Read-only.

For more information about returning a single object from a collection, see Returning an Object from a Collection.

Remarks

If there's more than one query running at the same time, the **ODBCErrors** collection contains the ODBC errors from the query that's finished last.

ODBCErrors Property Example

This example refreshes query table one and displays any ODBC errors that occur.

```
With Worksheets(1).QueryTables(1)
   .Refresh
   Set errs = Application.ODBCErrors
   If errs.Count > 0 Then
      Set r = .Destination.Cells(1)
      r.Value = "The following errors occurred:"
      c = 0
      For Each er In errs
         c = c + 1
         r.offset(c, 0).value = er.ErrorString
         r.offset(c, 1).value = er.SqlState
      Next
   Else
      MsgBox "Query complete: all records returned."
   End If
End With
```

ODBCTimeout Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproODBCTimeoutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproODBCTimeoutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproODBCTimeoutA"}

Returns or sets the ODBC query time limit, in seconds. The default value is 45 seconds. Read/write **Long**.

Remarks

The value 0 (zero) indicates an indefinite time limit.

ODBCTimeout Property Example

This example sets the ODBC query time limit to 15 seconds.

```
Application.ODBCTimeout = 15
```

Parameters Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproParametersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproParametersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproParametersA"}

Returns a **Parameters** collection that represents the query table parameters. Read-only.

For more information about returning a single object from a collection, see Returning an Object from a Collection.

Parameters Property Example

This example returns the **Parameters** collection from an existing parameter query. If the first parameter uses the character data type, the user is instructed to enter characters only in the prompt dialog box.

```
With Sheets("sheet1").QueryTables(1).Parameters(1)
    If .DataType = xlParamTypeVarChar Then
        .SetParam xlPrompt, "Enter a character only"
    End If
End With
```

PostText Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPostTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPostTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPostTextA"}

Returns or sets the string used with the post method of inputting data into a Web server to return data from a Web query. Read/write **String**.

Remarks

Microsoft Excel includes sample Web queries that you can modify by changing the HTML code by using WordPad or another text editor. You can find these samples in the Queries folder where you installed Microsoft Office.

The *Microsoft Office 97 Resource Kit* contains information about how to create a Web query. For information about how to obtain the Office Resource Kit, click  .

QueryTable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproQueryTableC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproQueryTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproQueryTableA"}

Returns a **QueryTable** object that represents the query table that intersects the specified range. Read-only.

QueryTable Property Example

This example refreshes the query table that intersects cell A10 on worksheet one.

```
Worksheets(1).Range("a10").QueryTable.Refresh
```

QueryTables Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproQueryTablesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproQueryTablesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproQueryTablesA"}

Returns the **QueryTables** collection that represents all the query tables on the specified worksheet. Read-only.

For more information about returning a single object from a collection, see Returning an Object from a Collection.

QueryTables Property Example

This example refreshes all query tables on worksheet one.

```
For Each qt in Worksheets(1).QueryTables
   qt.Refresh
Next
```

This example sets query table one so that formulas to the right of it are automatically updated whenever it's refreshed.

```
Sheets("sheet1").QueryTables(1).FillAdjacentFormulas = True
```

Recordset Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRecordsetC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRecordsetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRecordsetA"}

Returns or sets a **Recordset** object that's used as the data source for the specified query table. Read/write.

Remarks

If this property is used to overwrite an existing recordset, the change takes effect when the **Refresh** method is run.

Recordset Property Example

This example changes the **Recordset** object used with query table one and then refreshes the query table.

```
With Worksheets(1).QueryTables(1)
    .Recordset = OpenDatabase("c:\Nwind.mdb").OpenRecordset("employees")
    .Refresh
End With
```

Refreshing Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefreshingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefreshingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefreshingA"}

True if there's a background query in progress for the specified query table. Read/write **Boolean**.

Remarks

Use the **CancelRefresh** method to cancel background queries.

Refreshing Property Example

This example displays a message box if there's a background query in progress for query table one.

```
With Worksheets(1).QueryTables(1)
    If .Refreshing Then
        MsgBox "Query is currently refreshing: please wait"
    Else
        .Refresh BackgroundQuery := False
        .ResultRange.Select
    End If
End With
```

RefreshStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRefreshStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRefreshStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRefreshStyleA"}

Returns or sets the way rows on the specified worksheet are added or deleted to accommodate the number of rows in a recordset returned by a query. Read/write **Long**.

Can be one of the following **xlCellInsertionMode** constants.

Constant	Description
xlOverwriteCells	No new cells or rows are added to the worksheet. Data in surrounding cells is overwritten to accommodate any overflow.
xlInsertDeleteCells	Partial rows are inserted or deleted to match the exact number of rows required for the new recordset.
xlInsertEntireRows	Entire rows are inserted, if necessary, to accommodate any overflow. No cells or rows are deleted from the worksheet.

RefreshStyle Property Example

This example adds a query table to Sheet1. The **RefreshStyle** property adds rows to the worksheet as needed, to hold the data results.

```
Dim qt As QueryTable
Set qt = Sheets("sheet1").QueryTables _
    .Add(Connection:="Finder;c:\myfile.dqy", _
      Destination:=Range("sheet1!a1"))
With qt
    .RefreshStyle = xlInsertEntireRows
    .Refresh
End With
```

ResultRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproResultRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproResultRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproResultRangeA"}

Returns a **Range** object that represents the area of the worksheet occupied by the specified query table. Read-only.

Remarks

The range doesn't include the field name row or the row number column.

ResultRange Property Example

This example sums the data in the first column of query table one. The sum of the first column is displayed below the data range.

```
Set c1 = Sheets("sheet1").QueryTables(1).ResultRange.Columns(1)
c1.Name = "Column1"
c1.End(xlDown).Offset(2, 0).Formula = "=sum(Column1)"
```

RowNumbers Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproRowNumbersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproRowNumbersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproRowNumbersA"}

True if row numbers are added as the first column of the specified query table. Read/write **Boolean**.

Remarks

Setting this property to **True** doesn't immediately cause row numbers to appear. The row numbers appear the next time the query table is refreshed, and they're reconfigured every time the query table is refreshed.

RowNumbers Property Example

This example adds row numbers and field names to the query table.

```
With Worksheets(1).QueryTables("ExternalData1")
    .RowNumbers = True
    .FieldNames = True
    .Refresh
End With
```

SavePassword Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSavePasswordC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSavePasswordX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSavePasswordA"}

True if password information in an ODBC connection string is saved with the specified query. **False** if the password is removed. Read/write **Boolean**.

Remarks

This property affects only ODBC queries.

SavePassword Property Example

This example causes password information to be removed from the ODBC connection string whenever query table one is saved.

```
Worksheets(1).QueryTables(1).SavePassword = False
```

Sql Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSqlC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSqlX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSqlA"}

Returns or sets the SQL query string used with the specified ODBC data source. Read/write **String**.

Remarks

This property supports the full ODBC Data Manipulation Language (DML) grammar, including wild card characters, and stored procedures that return data. This property doesn't support Data Definition Language (DDL) statements.

Sql Property Example

This example changes the SQL query string for query tabel one and then refreshes the query table.

```
With Worksheets(1).QueryTables(1)
    .Sql = "select 96Sales.totals from 96Sales where profit < 5"
    .Refresh
End With
```

SqlState Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSqlStateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSqlStateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSqlStateA"}

Returns the SQL state error. Read-only **String**.

Remarks

For an explanation of the specific error, see you SQL documentation.

SqlState Property Example

This example refreshes query table one and displays any ODBC errors that occur.

```
With Worksheets(1).QueryTables(1)
   .Refresh
   Set errs = Application.ODBCErrors
   If errs.Count > 0 Then
      Set r = .Destination.Cells(1)
      r.Value = "The following errors occurred:"
      c = 0
      For Each er In errs
         c = c + 1
         r.offset(c, 0).value = er.ErrorString
         r.offset(c, 1).value = er.SqlState
      Next
   Else
      MsgBox "Query complete: all records returned."
   End If
End With
```

PromptString Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproPromptStringC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproPromptStringX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproPromptStringA"}

Returns the phrase that prompts the user for a parameter value in a parameter query. Read-only **String**.

PromptString Property Example

This example modifies the parameter prompt string for query table one.

```
With Worksheets(1).QueryTables(1).Parameters(1)
    .SetParam xlPrompt, "Please " & .PromptString
End With
```

SetParam Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"XLmthSetParamC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"XLmthSetParamX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"XLmthSetParamA"}

Defines a parameter for the specified query table.

Syntax

expression.**SetParam(*Type, Value*)**

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

Type Required **Long**. The parameter type. Can be one of the following **XlParameterType** constants.

Constant	Description
xlConstant	Uses the value specified by the ***Value*** argument.
xlPrompt	Displays a dialog box that prompts the user for the value. The ***Value*** argument specifies the text shown in the dialog box.
xlRange	Uses the value of the cell in the upper-left corner of the range. The ***Value*** argument specifies a **Range** object.

Value Required **Variant**. The value of the specified parameter, as shown in the description of the ***Type*** argument.

SetParam Method Example

This example changes the SQL statement for query table one. The clause "(city=?)" indicates that the query is a parameter query, and the example sets the value of city to the constant "Oakland."

```
Set qt = Sheets("sheet1").QueryTables(1)
qt.Sql = "SELECT * FROM authors  WHERE (city=?)"
Set param1 = qt.Parameters.Add("City Parameter", xlParamTypeVarChar)
param1.SetParam xlConstant, "Oakland"
qt.Refresh
```

This example sets the value of city to the value of cell A2 on worksheet two.

```
Set qt = Sheets("sheet1").QueryTables(1)
qt.Sql = "SELECT * FROM authors  WHERE (city=?)"
Set param1 = qt.Parameters.Add("City Parameter", xlParamTypeVarChar)
param1.SetParam xlRange, Range("sheet2!a1")
qt.Refresh
```

SourceRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproSourceRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproSourceRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproSourceRangeA"}

Returns a **Range** object that represents the cell that contains the value of the specified query parameter. Read-only.

SourceRange Property Example

This example changes the value of the cell used as the source range for the query.

```
Set qt = Sheets("sheet1").QueryTables(1)
Set param1 = qt.Parameters(1)
Set r = param1.SourceRange
r.Value = "New York"
qt.Refresh
```

Add Method (Parameters Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"XLmthAddParametersObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"XLmthAddParametersObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"XLmthAddParametersObjA"}

Creates a new query parameter. Returns a **Parameter** object that represents the new query parameter.

Syntax

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

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

Name Required **String**. The name of the specified parameter. The parameter name should match the parameter clause in the SQL statement.

DataType Optional **Variant**. The data type of the parameter. Can be one of the following **XlParameterDataType** constants:

xlParamTypeBigInt	**xlParamTypeNumeric**
xlParamTypeBinary	**xlParamTypeLongVarChar**
xlParamTypeBit	**xlParamTypeReal**
xlParamTypeChar	**xlParamTypeSmallInt**
xlParamTypeDate	**xlParamTypeTime**
xlParamTypeDecimal	**xlParamTypeTimeStamp**
xlParamTypeDouble	**xlParamTypeTinyInt**
xlParamTypeFloat	**xlParamTypeUnknown**
xlParamTypeInteger	**xlParamTypeVarBinary**
xlParamTypeLongVarBinary	**xlParamTypeVarChar**

These values correspond to ODBC data types. They indicate the type of value the ODBC driver is expecting to receive. Microsoft Excel and the ODBC driver manager will coerce the parameter value given in Microsoft Excel into the correct data type for the driver.

Add Method (Parameters Collection) Example

This example changes the SQL statement for query table one. The clause "(city=?)" indicates that the query is a parameter query, and the value of city is set to the constant "Oakland."

```
Set qt = Sheets("sheet1").QueryTables(1)
qt.Sql = "SELECT * FROM authors  WHERE (city=?)"
Set param1 = qt.Parameters.Add("City Parameter", xlParamTypeVarChar)
param1.SetParam xlConstant, "Oakland"
qt.Refresh
```

RefreshAll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"XLmthRefreshAllC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"XLmthRefreshAllX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"XLmthRefreshAllA"}

Refreshes all query tables and PivotTables in the specified workbook.

Syntax

expression.**RefreshAll**

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

RefreshAll Method Example

This example refreshes all the query tables and PivotTables in workbook three. The query tables and PivotTables that have the **BackgroundQuery** property set to **True** are refreshed in the background.

```
Workbooks(3).RefreshAll
```

TablesOnlyFromHTML Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproTablesOnlyFromHTMLC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproTablesOnlyFromHTMLX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproTablesOnlyFromHTMLA"}

True if only the HTML tables in the document are read when a query table is refreshed. **False** if the entire HTML document is read when a query table is refreshed. This property has an effect only when the query table is using a URL connection and the Web query returns an HTML document. Read/write **Boolean**.

TablesOnlyFromHTML Property Example

This example sets query table one to update only tables from an HTML document..

```
Worksheets(1).QueryTables(1).TablesOnlyFromHTML = True
```

EnableEditing Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlproEnableEditingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlproEnableEditingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"xlproEnableEditingA"}

True if the user can edit the specified query table. **False** if the user can only refresh the query table. Read/write **Boolean**.

EnableEditing Property Example

This example sets query table one so that the user cannot edit it.

```
Worksheets(1).QueryTables(1).EnableEditing = False
```

Parameter Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjParameterC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjParameterP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjParameterM"}

Worksheets (Worksheet)

└

QueryTables (QueryTable)

└

Parameters (Parameter)

Represents a single parameter used in a parameter query. The **Parameter** object is a member of the **Parameters** collection.

Using the Parameter Object

Use **Parameters**(*index*), where *index* is the index number of the parameter, to return a single **Parameter** object. The following example modifies the prompt string for parameter one.

```
With Worksheets(1).QueryTables(1).Parameters(1)
    .SetParam xlPrompt, "Please " & .PromptString
End With
```

Parameters Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjParametersC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjParametersP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjParametersM"}

A collection of **Parameter** objects for the specified query table. Each **Parameter** object represents a single query parameter. Every query table contains a **Parameters** collection, but the collection is empty unless the query table is using a parameter query.

Using the Parameters Collection

Use the **Parameters** property to return the **Parameters** collection. The following example displays the number of parameters in query table one.

```
MsgBox Workbooks(1).ActiveSheet.QueryTables(1).Parameters.Count
```

Use the **Add** method to create a new parameter for a query table. The following example changes the SQL statement for query table one. The clause "(city=?)" indicates that the query is a parameter query, and the value of city is set to the constant "Oakland."

```
Set qt = Sheets("sheet1").QueryTables(1)
qt.Sql = "SELECT * FROM authors  WHERE (city=?)"
Set param1 = qt.Parameters.Add("City Parameter", xlParamTypeVarChar)
param1.SetParam xlConstant, "Oakland"
qt.Refresh
```

You cannot use the **Add** method on a URL connection query table. For URL connection query tables, Microsoft Excel creates the parameters based on the **Connection** and **PostText** properties.

ODBCError Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjODBCErrorC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjODBCErrorP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjODBCErrorM"}

Application

└

ODBCErrors (ODBCError)

Represents an ODBC error generated by the the most recent ODBC query. The **ODBCError** object is a member of the **ODBCErrors** collection. If the specified ODBC query runs without error, the **ODBCErrors** collection is empty. The errors in the collection are indexed in the order in which they're generated by the ODBC data source.

Using the ODBCError Object

Use **ODBCErrors**(*index*), where *index* is the index number of the error, to return a single **ODBCError** object. The following example refreshes query table one and displays the first ODBC error that occurs.

```
With Worksheets(1).QueryTables(1)
   .Refresh
   If Application.ODBCErrors.Count > 0 Then
      Set er = Application.ODBCErrors(1)
      MsgBox "The following error occurred:" &
         er.ErrorString & " : " & er.SqlState
   Else
      MsgBox "Query complete: all records returned."
   End If
End With
```

ODBCErrors Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjODBCErrorsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjODBCErrorsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjODBCErrorsM"}

Application

└

ODBCErrors (ODBCError)

A collection of **ODBCError** objects. Each **ODBCError** object represents an error returned by the most recent ODBC query. If the specified ODBC query runs without error, the **ODBCErrors** collection is empty. The errors in the collection are indexed in the order in which they're generated by the ODBC data source. You cannot add members to the collection.

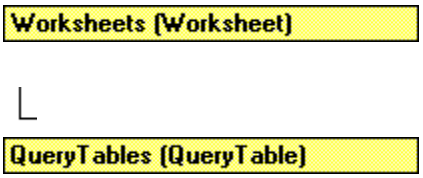
Using the ODBCErrors Collection

Use the **ODBCErrors** property to return the **ODBCErrors** collection. The following example refreshes query table one and displays any ODBC errors that occur.

```
With Worksheets(1).QueryTables(1)
   .Refresh
   Set errs = Application.ODBCErrors
   If errs.Count > 0 Then
      Set r = .Destination.Cells(1)
      r.Value = "The following errors occurred:"
      c = 0
      For Each er In errs
         c = c + 1
         r.offset(c, 0).value = er.ErrorString
         r.offset(c, 1).value = er.SqlState
      Next
   Else
      MsgBox "Query complete: all records returned."
   End If
End With
```

QueryTable Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjQueryTableC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjQueryTableP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjQueryTableM"}



Represents a worksheet table built from data returned from an external data source, such as an SQL server or a Microsoft Access database. The **QueryTable** object is a member of the **QueryTables** collection.

Using the QueryTable Object

Use **QueryTables**(*index*), where *index* is the index number of the query table, to return a single **QueryTable** object. The following example sets query table one so that formulas to the right of it are automatically updated whenever it's refreshed.

```
Sheets("sheet1").QueryTables(1).FillAdjacentFormulas = True
```

QueryTables Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjQueryTablesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjQueryTablesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjQueryTablesM"}

A collection of **QueryTable** objects. Each **QueryTable** object represents a worksheet table built from data returned from an external data source.

Using the QueryTables Collection

Use the **QueryTables** property to return the **QueryTables** collection. The following example displays the number of query tables on the active worksheet.

```
MsgBox ActiveSheet.QueryTables.Count
```

Use the **Add** method to create a new query table and add it to the **QueryTables** collection. The following example creates a new query table.

```
Dim qt As QueryTable
sqlstring = "select 96Sales.totals from 96Sales where profit < 5"
connstring = _
   "ODBC;DSN=96SalesData;UID=Rep21;PWD=NUyHwYQI;Database=96Sales"
With ActiveSheet.QueryTables.Add(Connection:=connstring, _
      Destination:=Range("B1"), Sql:=sqlstring)
   .Refresh
End With
```

AddIn Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAddInC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjAddInX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAddInP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAddInM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjAddInE"}

Application

AddIns (AddIn)

Represents a single add-in, either installed or not installed. The **AddIn** object is a member of the **AddIns**collection. The **AddIns** collection contains a list of all the add-ins available to Microsoft Excel, regardless of whether they're installed. 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 title or index number, to return a single **AddIn** object. The following example installs the Analysis Toolpak add-in.

```
AddIns("analysis toolpak").Installed = 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 creates a list that contains specified properties of the available add-ins.

```
With Worksheets("sheet1")
    .Rows(1).Font.Bold = True
    .Range("a1:d1").Value = _
       Array("Name", "Full Name", "Title", "Installed")
    For i = 1 To AddIns.Count
       .Cells(i + 1, 1) = AddIns(i).Name
       .Cells(i + 1, 2) = AddIns(i).FullName
       .Cells(i + 1, 3) = AddIns(i).Title
       .Cells(i + 1, 4) = AddIns(i).Installed
    Next
    .Range("a1").CurrentRegion.Columns.AutoFit
End With
```

Remarks

The **Add** method adds an add-in to the list of available add-ins but doesn't install the add-in. Set the **Installed** property of the add-in to **True** to install the add-in. To install an add-in that doesn't appear in the list of available add-ins, you must first use the **Add** method and then set the **Installed** property. This can be done 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.xll").Installed = True
```

Use **Workbooks**(*index*) where *index* is the add-in filename (not title) to return a reference to the workbook corresponding to a loaded add-in. You must use the file name because loaded add-ins don't normally appear in the **Workbooks** collection. This example sets the `wb` variable to the workbook for Myaddin.xla.

```
Set wb = Workbooks("myaddin.xla")
```

The following example sets the wb variable to the workbook for the Analysis Toolpak add-in.

```
Set wb = Workbooks(AddIns("analysis toolpak").Name)
```

If the **Installed** property returns **True**, but calls to functions in the add-in still fail, the add-in may not actually be loaded. This is because the **Addin** object represents the existence and installed state of the add-in but doesn't represent the actual contents of the add-in workbook.To guarantee that an installed add-in is loaded, you should open the add-in workbook. The following example opens the workbook for the add-in named "My Addin" if the add-in isn't already present in the **Workbooks** collection.

```
On Error Resume Next ' turn off error checking
Set wbMyAddin = Workbooks(Addins("My Addin").Name)
lastError = Err
On Error Goto 0   ' restore error checking
If lastError <> 0 Then
   ' the add-in workbook isn't currently open. Manually open it.
   Set wbMyAddin = Workbooks.Open(Addins("My Addin").FullName)
End If
```

AddIns Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAddInsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjAddInsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAddInsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAddInsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjAddInsE"}

Application

AddIns (AddIn)

A collection of **AddIn** objects that represents all the add-ins available to Microsoft Excel, regardless of whether they're installed. 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 creates a list that contains the names and installed states of all the available add-ins.

```
Sub DisplayAddIns()
    Worksheets("Sheet1").Activate
    rw = 1
    For Each ad In Application.AddIns
        Worksheets("Sheet1").Cells(rw, 1) = ad.Name
        Worksheets("Sheet1").Cells(rw, 2) = ad.Installed
        rw = rw + 1
    Next
End Sub
```

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 install the add-in. Set the **Installed** property of the add-in to **True** to install the add-in. To install an add-in that doesn't appear in the list of available add-ins, you must first use the **Add** method and then set the **Installed** property. This can be done 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.xll").Installed = True
```

Use **AddIns**(*index*) where *index* is the add-in title or index number to return a single **AddIn** object. The following example installs the Analysis Toolpak add-in.

```
AddIns("analysis toolpak").Installed = 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.

Application Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjApplicationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjApplicationX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjApplicationP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjApplicationM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjApplicationE"}

Multiple Objects

Represents the entire Microsoft Excel application. The **Application** object contains:

- Application-wide settings and options (many of the options in the **Options** dialog box (**Tools** menu), for example).
- Methods that return top-level objects, such as **ActiveCell**, **ActiveSheet**, 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("book1.xls").Activate
```

The following example creates a Microsoft Excel worksheet object in another application and then opens a workbook in Microsoft Excel.

```
Set xl = CreateObject("Excel.Sheet")
xl.Application.Workbooks.Open "newbook.xls"
```

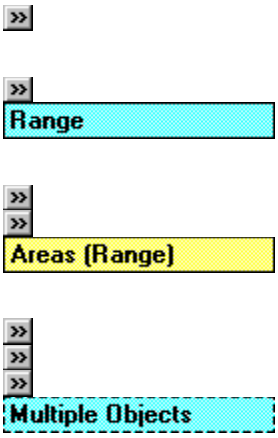
Remarks

Many of the properties and methods that return the most common user-interface objects, such as the active cell (**ActiveCell** property), can be used without the **Application** object qualifier. For example, instead of writing `Application.ActiveCell.Font.Bold = True`, you can write `ActiveCell.Font.Bold = True`.

AddIn, **Assistant**, **AutoCorrect**, **Chart**, **CommandBars**, **Debug**, **Dialog**, **RecentFiles**, **VBE**, **Window**, **Worksheet**, **WorksheetFunction**, **Workbook**

Areas Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAreasC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjAreasX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAreasP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAreasM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjAreasE"}



A collection of the areas, or contiguous blocks of cells, within a selection. There's no singular Area object; individual members of the **Areas** collection are **Range** objects. The **Areas** collection contains one **Range** object for each discrete, contiguous range of cells within the selection. If the selection contains only one area, the **Areas** collection contains a single **Range** object that corresponds to that selection.

Using the Areas Collection

Use the **Areas** property to return the **Areas** collection. The following example clears the current selection if it contains more than one area.

```
If Selection.Areas.Count <> 1 Then Selection.Clear
```

Use **Areas**(*index*), where *index* is the area index number, to return a single **Range** object from the collection. The index numbers correspond to the order in which the areas were selected. The following example clears the first area in the current selection if the selection contains more than one area.

```
If Selection.Areas.Count <> 1 Then
   Selection.Areas(1).Clear
End If
```

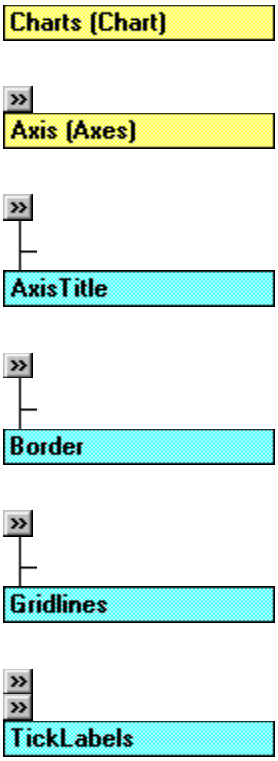
Some operations cannot be performed on more than one area in a selection at the same time; you must loop through the individual areas in the selection and perform the operations on each area separately. The following example performs the operation named "myOperation" on the selected range if the selection contains only one area; if the selection contains multiple areas, the example performs myOperation on each individual area in the selection.

```
Set rangeToUse = Selection
If rangeToUse.Areas.Count = 1 Then
    myOperation rangeToUse
Else
    For Each singleArea in rangeToUse.Areas
        myOperation singleArea
    Next
End If
```

Border, **Characters**, **Font**, **Interior**, **Name**, **SoundNote**, **Style**

Axes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAxesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjAxesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAxesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAxesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjAxesE"}



A collection of all the **Axis** objects in the specified chart.

Using the Axes Collection

Use the **Axes** method to return the **Axes** collection. The following example displays the number of axes on embedded chart one on worksheet one.

```
With Worksheets(1).ChartObjects(1).Chart
    MsgBox .Axes.Count
End With
```

Use **Axes**(*type*, *group*), where *type* is the axis type and *group* is the axis group, to return a single **Axis** object. *Type* can be one of the following **XlAxisType** constants: **xlCategory**, **xlSeries**, or **xlValue**. *Group* can be one of the following **XlAxisGroup** constants: **xlPrimary** or **xlSecondary**. For more information, see the **Axes** method.

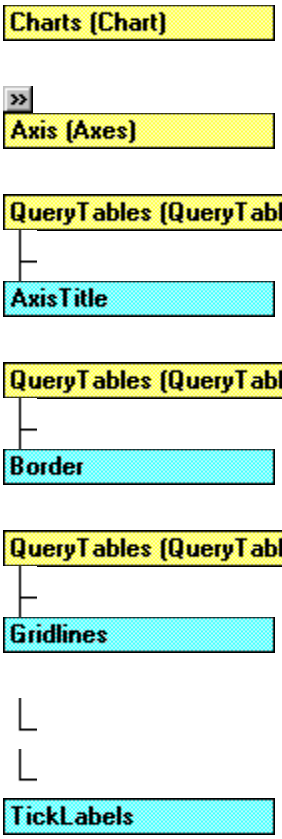
The following example sets the category axis title text on the chart sheet named "Chart1."

```
With Charts("chart1").Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Caption = "1994"
End With
```

Axis Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAxisC"}
{ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAxisP"}
{ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjAxisE"}

{ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjAxisX":1}
{ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAxisM"}



Represents a single axis in a chart. The **Axis** object is a member of the **Axes** collection.

Using the Axis Object

Use **Axes**(*type*, *group*) where *type* is the axis type and *group* is the axis group to return a single **Axis** object. *Type* can be one of the following **XlAxisType** constants: **xlCategory**, **xlSeries**, or **xlValue**. *Group* can be one of the following **XlAxisGroup** constants: **xlPrimary** or **xlSecondary**. For more information, see the **Axes** method.

The following example sets the category axis title text on the chart sheet named "Chart1."

```
With Charts("chart1").Axes(xlCategory)
    .HasTitle = True
    .AxisTitle.Caption = "1994"
End With
```

AxisTitle Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAxisTitleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjAxisTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAxisTitleP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAxisTitleM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjAxisTitleE"}

Charts (Chart)

AxisTitle

Represents a chart axis title.

Using the AxisTitle Object

Use the **AxisTitle** property to return an **AxisTitle** object. The following example activates embedded chart one, sets the value axis title text, sets the font to Bookman 10 point, and formats the word "millions" as italic.

```
Worksheets("sheet1").ChartObjects(1).Activate
With ActiveChart.Axes(xlValue)
    .HasTitle = True
    With .AxisTitle
        .Caption = "Revenue (millions)"
        .Font.Name = "bookman"
        .Font.Size = 10
        .Characters(10, 8).Font.Italic = True
    End With
End With
```

Remarks

The **AxisTitle** object doesn't exist and cannot be used unless the **HasTitle** property for the axis is **True**.

Border, **Characters**, **Font**, **Interior**

Border Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjBorderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjBorderX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjBorderP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjBorderM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjBorderE"}

Border

Represents the border of an object.

Using the Border Object

Most bordered objects (all except for the **Range** and **Style** objects) have a border that's treated as a single entity, regardless of how many sides it has. The entire border must be returned as a unit. Use the **Border** property to return the **Border** object for this kind of object. The following example activates the chart sheet named "Chart1," places a dashed border around the chart area for the active chart, and places a dotted border around the plot area.

```
Charts("chart1").Activate
With ActiveChart
    .ChartArea.Border.LineStyle = xlDash
    .PlotArea.Border.LineStyle = xlDot
End With
```

Range and **Style** objects have four discrete borders – left, right, top, and bottom – which can be returned individually or as a group. Use the **Borders** property to return the **Borders** collection, which contains all four borders. The following example adds a double border to cell A1 on worksheet one.

```
Worksheets(1).Range("a1").Borders.LineStyle = xlBorderStyleDouble
```

Use **Borders**(*index*), where *index* identifies the border, to return a single **Border** object. The following example sets the color of the bottom border of cells A1:G1.

```
Worksheets("Sheet1").Range("a1:g1").
    Borders(xlEdgeBottom).Color = RGB(255, 0, 0)
```

Index can be one of the following **XlBorderType** constants: **xlInsideHorizontal**, **xlInsideVertical**, **xlDiagonalDown**, **xlDiagonalUp**, **xlEdgeBottom**, **xlEdgeLeft**, **xlEdgeRight**, or **xlEdgeTop**.

Axis, **AxisTitle**, **ChartArea**, **ChartObject**, **ChartObjects**, **ChartTitle**, **DataLabel**, **DataLabels**, **DownBars**, **DropLines**, **ErrorBars**, **Floor**, **Gridlines**, **HiLoLines**, **Legend**, **LegendKey**, **OLEObject**, **OLEObjects**, **PlotArea**, **Point**, **Series**, **SeriesLines**, **Trendline**, **UpBars**, **Walls**

Borders Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjBordersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjBordersX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjBordersP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjBordersM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjBordersE"}

Border

A collection of four **Border** objects that represent the four borders of a **Range** or **Style** object.

Using the Borders Collection

Use the **Borders** property to return the **Borders** collection, which contains all four borders. The following example adds a double border to cell A1 on worksheet one.

```
Worksheets(1).Range("a1").Borders.LineStyle = xlBorderStyleDouble
```

Use **Borders**(*index*), where *index* identifies the border, to return a single **Border** object. The following example sets the color of the bottom border of cells A1:G1 to red.

```
Worksheets("Sheet1").Range("a1:g1"). _
    Borders(xlEdgeBottom).Color = RGB(255, 0, 0)
```

Index can be one of the following **XlBorderType** constants: **xlInsideHorizontal**, **xlInsideVertical**, **xlDiagonalDown**, **xlDiagonalUp**, **xlEdgeBottom**, **xlEdgeLeft**, **xlEdgeRight**, or **xlEdgeTop**.

Remarks

You can set border properties for an individual border only with **Range** and **Style** objects. Other bordered objects, such as check boxes and chart areas, have a border that's treated as a single entity, regardless of how many sides it has. For these objects, you must return and set properties for the entire border as a unit. For more information, see the **Border** object.

Range, **Style**

Characters Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCharactersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjCharactersX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCharactersP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCharactersM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCharactersE"}

Characters

Font

Represents characters in an object that contains text. The **Characters** object lets you modify any sequence of characters contained in the full text string.

Using the Characters Object

Use **Characters**(*start, length*), where *start* is the start character number and *length* is the number of characters, to return a **Characters** object. The following example adds text to cell B1 and then makes the second word bold.

```
With Worksheets("sheet1").Range("b1")
    .Value = "New Title"
    .Characters(5, 5).Font.Bold = True
End With
```

Remarks

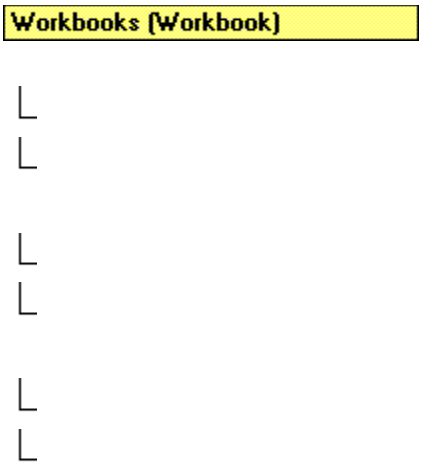
The **Characters** method is necessary only when you need to change some of an object's text without affecting the rest (you cannot use the **Characters** method to format a portion of the text if the object doesn't support rich text). To change all the text at the same time, you can usually apply the appropriate method or property directly to the object. The following example formats the contents of cell A5 as italic.

```
Worksheets("sheet1").Range("a5").Font.Italic = True
```

AxisTitle, **ChartTitle**, **DataLabel**, **Range**

Chart Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartE"}



Represents a chart in a workbook. The chart can be either an embedded chart (contained in a **ChartObject**) or a separate chart sheet.

Using the Chart Object

The following properties and methods for returning a **Chart** object are described in this section:

- **Chart** property
- **Charts** method
- **ActiveChart** property
- **ActiveSheet** property

Chart Property

Use the **Chart** property to return a **Chart** object that represents the chart contained in a **ChartObject** object. The following example sets the pattern for the chart area in embedded chart one on the worksheet named "Sheet1."

```
Worksheets("sheet1").ChartObjects(1).Chart. _
    ChartArea.Interior.Pattern = xlLightDown
```

Charts Method

The **Charts** collection contains a **Chart** object for each chart sheet in a workbook. Use **Charts**(*index*), where index is the chart-sheet index number or name, to return a single **Chart** object. The following example changes the color of series one on chart sheet one.

```
Charts(1).SeriesCollection(1).Interior.Color = RGB(255, 0, 0)
```

The chart index number represents the position of the chart sheet on the workbook tab bar. `Charts(1)` is the first (leftmost) chart in the workbook; `Charts(Charts.Count)` is the last (rightmost). All chart sheets are included in the index count, even if they're hidden. The chart-sheet name is shown on the workbook tab for the chart. You can use the **Name** property to set or return the chart name.

The following example moves the chart named "Sales" to the end of the active workbook.

```
Charts("sales").Move after:=Sheets(Sheets.Count)
```

The **Chart** object is also a member of the **Sheets** collection. The **Sheets** collection contains all the sheets in the workbook (both chart sheets and worksheets). Use **Sheets**(*index*), where *index* is the sheet index number or name, to return a single sheet.

ActiveChart Property

When a chart is the active object, you can use the **ActiveChart** property to refer to it. A chart sheet is active if the user has selected it or it's been activated with the **Activate** method. The following example activates chart sheet one and then sets the chart type and title.

```
Charts(1).Activate
With ActiveChart
   .Type = xlLine
   .HasTitle = True
   .ChartTitle.Text = "January Sales"
End With
```

An embedded chart is active if the user has selected it or the **ChartObject** object that it's contained in has been activated with the **Activate** method. The following example activates embedded chart one on worksheet one and then sets the chart type and title. Notice that after the embedded chart has been activated, the code in this example is the same as that in the previous example. Using the **ActiveChart** property allows you to write Visual Basic code that can refer to either an embedded chart or a chart sheet (whichever is active).

```
Worksheets(1).ChartObjects(1).Activate
ActiveChart.Type = xlLine
ActiveChart.HasTitle = True
ActiveChart.ChartTitle.Text = "January Sales"
```

ActiveSheet Property

When a chart sheet is the active sheet, you can use the **ActiveSheet** property to refer to it. The following example uses the **Activate** method to activate the chart sheet named "Chart1" and then sets the interior color for series one in the chart to blue.

```
Charts("chart1").Activate
ActiveSheet.SeriesCollection(1).Interior.ColorIndex = 5
```

Axis, **ChartArea**, **ChartGroups**, **ChartObjects**, **ChartTitle**, **Corners**, **Floor**, **Legend**, **OLEObjects**, **PageSetup**, **PlotArea**, **SeriesCollection**, **Walls**

ChartArea Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartAreaC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartAreaX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartAreaP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartAreaM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartAreaE"}

Represents the chart area of a chart. The chart area on a 2-D chart contains the axes, the chart title, the axis titles, and the legend. The chart area on a 3-D chart contains the chart title and the legend; it doesn't include the plot area (the area within the chart area where the data is plotted). For information about formatting the plot area, see the **PlotArea** object.

Using the ChartArea Object

Use the **ChartArea** property to return the **ChartArea** object. The following example sets the pattern for the chart area in embedded chart one on the worksheet named "Sheet1."

```
Worksheets("sheet1").ChartObjects(1).Chart. _
    ChartArea.Interior.Pattern = xlLightDown
```

ChartGroup Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartGroupC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartGroupP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartGroupM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartGroupE"}

ChartGroups (ChartGroup)

Represents one or more series plotted in a chart with the same format. A chart contains one or more chart groups, each chart group contains one or more series, and each series contains one or more points. For example, a single chart might contain both a line chart group, containing all the series plotted with the line chart format, and a bar chart group, containing all the series plotted with the bar chart format. The **ChartGroup** object is a member of the **ChartGroups** collection.

Using the ChartGroup Object

Use **ChartGroups**(*index*), where *index* is the chart-group index number, to return a single **ChartGroup** object. The following example adds drop lines to chart group one on chart sheet one.

```
Charts(1).ChartGroups(1).HasDropLines = True
```

If the chart has been activated, you can use the **ActiveChart** property.

```
Charts(1).Activate
ActiveChart.ChartGroups(1).HasDropLines = True
```

Because the index number for a particular chart group can change if the chart format used for that group is changed, it may be easier to use one of the named chart group shortcut methods to return a particular chart group. The **PieGroups** method returns the collection of pie chart groups in a chart, the **LineGroups** method returns the collection of line chart groups, and so on. Each of these methods can be used with an index number to return a single **ChartGroup** object, or without an index number to return a **ChartGroups** collection. The following chart group methods are available:

- **AreaGroups** method
- **BarGroups** method
- **ColumnGroups** method
- **DoughnutGroups** method
- **LineGroups** method
- **PieGroups** method

DownBars, **DropLines**, **HiLoLines**, **SeriesCollection**, **SeriesLines**, **UpBars**

ChartGroups Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartGroupsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartGroupsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartGroupsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartGroupsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartGroupsE"}

A collection of all the **ChartGroup** objects in the specified chart. Each **ChartGroup** object represents one or more series plotted in a chart with the same format. A chart contains one or more chart groups, each chart group contains one or more series, and each series contains one or more points. For example, a single chart might contain both a line chart group, containing all the series plotted with the line chart format, and a bar chart group, containing all the series plotted with the bar chart format.

Using the ChartGroups Collection

Use the **ChartGroups** method to return the **ChartGroups** collection. The following example displays the number of chart groups on embedded chart one on worksheet one.

```
MsgBox Worksheets(1).ChartObjects(1).Chart.ChartGroups.Count
```

Use **ChartGroups**(*index*), where *index* is the chart-group index number, to return a single **ChartGroup** object. The following example adds drop lines to chart group one on chart sheet one.

```
Charts(1).ChartGroups(1).HasDropLines = True
```

If the chart has been activated, you can use **ActiveChart**:

```
Charts(1).Activate
ActiveChart.ChartGroups(1).HasDropLines = True
```

Because the index number for a particular chart group can change if the chart format used for that group is changed, it may be easier to use one of the named chart group shortcut methods to return a particular chart group. The **PieGroups** method returns the collection of pie chart groups in a chart, the **LineGroups** method returns the collection of line chart groups, and so on. Each of these methods can be used with an index number to return a single **ChartGroup** object, or without an index number to return a **ChartGroups** collection. The following chart group methods are available:

- **AreaGroups** method
- **BarGroups** method
- **ColumnGroups** method
- **DoughnutGroups** method
- **LineGroups** method
- **PieGroups** method

ChartObject Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartObjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartObjectP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartObjectM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartObjectE"}

ChartObjects (ChartObject)

Chart

Interior

Represents an embedded chart on a worksheet. The **ChartObject** object acts as a container for a **Chart** object. Properties and methods for the **ChartObject** object control the appearance and size of the embedded chart on the worksheet. The **ChartObject** object is a member of the **ChartObjects** collection. The **ChartObjects** collection contains all the embedded charts on a single sheet.

Using the ChartObject Object

Use **ChartObjects**(*index*), where index is the embedded chart index number or name, to return a single **ChartObject** object. The following example sets the pattern for the chart area in embedded chart one on the worksheet named "Sheet1."

```
Worksheets("Sheet1").ChartObjects(1).Chart. _
    ChartArea.Interior.Pattern = xlLightDown
```

The embedded chart name is shown in the **Name** box when the embedded chart is selected. Use the **Name** property to set or return the name of the **ChartObject** object. The following example puts rounded corners on the embedded chart named "Chart 1" on the worksheet named "Sheet1."

```
Worksheets("sheet1").ChartObjects("chart 1").RoundedCorners = True
```

ChartObjects Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartObjectsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartObjectsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartObjectsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartObjectsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartObjectsE"}

A collection of all the **ChartObject** objects on the specified chart sheet, dialog sheet, or worksheet. Each **ChartObject** object represents an embedded chart. The **ChartObject** object acts as a container for a **Chart** object. Properties and methods for the **ChartObject** object control the appearance and size of the embedded chart on the sheet.

Using the ChartObjects Collection

Use the **ChartObjects** method to return the **ChartObjects** collection. The following example deletes all the embedded charts on the worksheet named "Sheet1."

```
Worksheets("sheet1").ChartObjects.Delete
```

Use the **Add** method to create a new, empty embedded chart and add it to the collection. Use the **ChartWizard** method to add data and format the new chart. The following example creates a new embedded chart and then adds the data from cells A1:A20 as a line chart.

```
Dim ch As ChartObject
Set ch = Worksheets("sheet1").ChartObjects.Add(100, 30, 400, 250)
ch.Chart.ChartWizard source:=Worksheets("sheet1").Range("a1:a20"), _
    gallery:=xlLine, title:="New Chart"
```

Use **ChartObjects**(*index*), where index is the embedded chart index number or name, to return a single **ChartObject** object. The following example sets the pattern for the chart area in embedded chart one on the worksheet named "Sheet1."

```
Worksheets("Sheet1").ChartObjects(1).Chart. _
    ChartArea.Interior.Pattern = xlLightDown
```

Charts Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartsE"}

Workbooks (Workbook)

A collection of all the chart sheets in the specified or active workbook. Each chart sheet is represented by a **Chart** object. This doesn't include charts embedded on worksheets or dialog sheets. For information about embedded charts, see the **Chart** or **ChartObject** object.

Using the Charts Collection

Use the **Charts** property to return the **Charts** collection. The following example prints all chart sheets in the active workbook.

```
Charts.PrintOut
```

Use the **Add** method to create a new chart sheet and add it to the workbook. The following example adds a new chart sheet to the active workbook and places the new chart sheet immediately after the worksheet named "Sheet1."

```
Charts.Add after:=Worksheets("sheet1")
```

You can combine the **Add** method with the **ChartWizard** method to add a new chart that contains data from a worksheet. The following example adds a new line chart based on data in cells A1:A20 on the worksheet named "Sheet1."

```
With Charts.Add
    .ChartWizard source:=Worksheets("sheet1").Range("a1:a20"), _
     gallery:=xlLine, title:="February Data"
End With
```

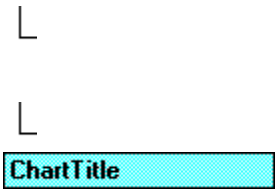
Use **Charts**(*index*), where index is the chart-sheet index number or name, to return a single **Chart** object. The following example changes the color of series one on chart sheet one to red.

```
Charts(1).SeriesCollection(1).Interior.Color = RGB(255, 0, 0)
```

The **Sheets** collection contains all the sheets in the workbook (both chart sheets and worksheets). Use **Sheets**(*index*), where *index* is the sheet name or number, to return a single sheet.

ChartTitle Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartTitleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjChartTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartTitleP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartTitleM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartTitleE"}



Represents the chart title.

Using the ChartTitle Object

Use the **ChartTitle** property to return the **ChartTitle** object. The following example adds a title to embedded chart one on the worksheet named "Sheet1."

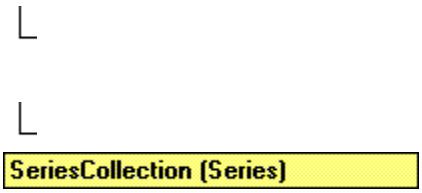
```
With Worksheets("sheet1").ChartObjects(1).Chart
    .HasTitle = True
    .ChartTitle.Text = "February Sales"
End With
```

Remarks

The **ChartTitle** object doesn't exist and cannot be used unless the **HasTitle** property for the chart is **True**.

DataLabel Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjDataLabelC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjDataLabelX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjDataLabelP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjDataLabelM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjDataLabelE"}



Represents the data label on a chart point or trendline. On a series, the **DataLabel** object is a member of the **DataLabels** collection. The **DataLabels** collection contains a **DataLabel** object for each point. For a series without definable points (such as an area series), the **DataLabels** collection contains a single **DataLabel** object.

Using the DataLabel Object

Use **DataLabels**(*index*), where *index* is the data-label index number, to return a single **DataLabel** object. The following example sets the number format for the fifth data label in series one in embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Chart _
    .SeriesCollection(1).DataLabels(5).NumberFormat = "0.000"
```

Use the **DataLabel** property to return the **DataLabel** object for a single point. The following example turns on the data label for the second point in series one on the chart sheet named "Chart1" and sets the data label text to "Saturday."

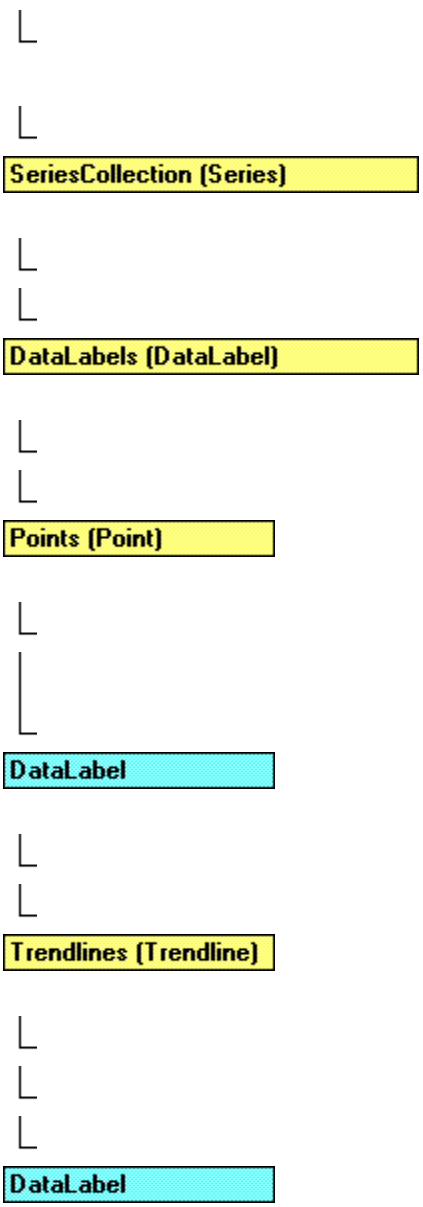
```
With Charts("chart1")
    With .SeriesCollection(1).Points(2)
        .HasDataLabel = True
        .DataLabel.Text = "Saturday"
    End With
End With
```

On a trendline, the **DataLabel** property returns the text shown with the trendline. This can be the equation, the R-squared value, or both (if both are showing). The following example sets the trendline text to show only the equation and then places the data label text in cell A1 on the worksheet named "Sheet1."

```
With Charts("chart1").SeriesCollection(1).Trendlines(1)
    .DisplayRSquared = False
    .DisplayEquation = True
    Worksheets("sheet1").Range("a1").Value = .DataLabel.Text
End With
```

DataLabels Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjDataLabelsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjDataLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjDataLabelsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjDataLabelsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjDataLabelsE"}



A collection of all the **DataLabel** objects for the specified series. Each **DataLabel** object represents a data label for a point or trendline. For a series without definable points (such as an area series), the **DataLabels** collection contains a single data label.

Using the Datalabels Collection

Use the **DataLabels** method to return the **DataLabels** collection. The following example sets the number format for data labels on series one on chart sheet one.

```
With Charts(1).SeriesCollection(1)
    .HasDataLabels = True
```

```
    .DataLabels.NumberFormat = "##.##"
End With
```

Use **DataLabels**(*index*), where *index* is the data-label index number, to return a single **DataLabel** object. The following example sets the number format for the fifth data label in series one in embedded chart one on worksheet one.

```
Worksheets(1).ChartObjects(1).Chart _
   .SeriesCollection(1).DataLabels(5).NumberFormat = "0.000"
```

Dialog Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjDialogC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjDialogX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjDialogP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjDialogM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjDialogE"}

Dialogs (Dialog)

Represents a built-in Microsoft Excel dialog box. The **Dialog** object is a member of the **Dialogs** collection. The **Dialogs** collection contains all the built-in dialog boxes in Microsoft Excel. You cannot create a new built-in dialog box or add one to the collection. The only useful thing you can do with a **Dialog** object is use it with the **Show** method to display the corresponding dialog box.

Using the Dialog Object

Use **Dialogs**(*index*), where *index* is a built-in constant identifying the dialog box, to return a single **Dialog** object. The following example runs the built-in **Open** dialog box (**File** menu). The **Show** method returns **True** if Microsoft Excel successfully opens a file; it returns **False** if the user cancels the dialog box.

```
dlgAnswer = Application.Dialogs(xlDialogOpen).Show
```

The Microsoft Excel Visual Basic object library includes built-in constants for many of the built-in dialog boxes. Each constant is formed from the prefix "xlDialog" followed by the name of the dialog box. For example, the **Apply Names** dialog box constant is **xlDialogApplyNames**, and the **Find File** dialog box constant is **xlDialogFindFile**. These constants are members of the **XlBuiltinDialog** enumerated type. For more information about the available constants, see Built-in Dialog Box Argument Lists.

Dialogs Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjDialogsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjDialogsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjDialogsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjDialogsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjDialogsE"}

Dialogs (Dialog)

A collection of all the **Dialog** objects in Microsoft Excel. Each **Dialog** object represents a built-in dialog box. You cannot create a new built-in dialog box or add one to the collection. The only useful thing you can do with a **Dialog** object is use it with the **Show** method to display the dialog corresponding dialog box.

Using the Dialogs Collection

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

```
MsgBox Application.Dialogs.Count
```

Use **Dialogs**(*index*), where *index* is a built-in constant identifying the dialog box, to return a single **Dialog** object. The following example runs the built-in **File Open** dialog box.

```
dlgAnswer = Application.Dialogs(xlDialogOpen).Show
```

The Microsoft Excel Visual Basic object library includes built-in constants for many of the built-in dialog boxes. Each constant is formed from the prefix "xlDialog" followed by the name of the dialog box. For example, the **Apply Names** dialog box constant is **xlDialogApplyNames**, and the **Find File** dialog box constant is **xlDialogFindFile**. These constants are members of the **XlBuiltinDialog** enumerated type. For more information about the available constants, see Built-in Dialog Box Argument Lists.

DownBars Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjDownBarsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjDownBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjDownBarsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjDownBarsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjDownBarsE"}

Represents the down bars in a chart group. Down bars connect points on the first series in the chart group with lower values on the last series (the lines go down from the first series). Only 2-D line groups that contain at least two series can have down bars. This object isn't a collection. There's no object that represents a single down bar; you either have up bars and down bars turned on for all points in a chart group or you have them turned off.

Using the DownBars Object

Use the **DownBars** property to return the **DownBars** object. The following example turns on up and down bars for chart group one in embedded chart one on the worksheet named "Sheet5." The example then sets the up bar color to blue and the down bar color to red.

```
With Worksheets("sheet5").ChartObjects(1).Chart.ChartGroups(1)
    .HasUpDownBars = True
    .UpBars.Interior.Color = RGB(0, 0, 255)
    .DownBars.Interior.Color = RGB(255, 0, 0)
End With
```

Remarks

If the **HasUpDownBars** property is **False**, most properties of the **DownBars** object are disabled.

DropLines Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjDropLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjDropLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjDropLinesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjDropLinesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjDropLinesE"}

Represents the drop lines in a chart group. Drop lines connect the points in the chart with the x-axis. Only line and area chart groups can have drop lines. This object isn't a collection. There's no object that represents a single drop line; you either have drop lines turned on for all points in a chart group or you have them turned off.

Using the DropLines Object

Use the **DropLines** property to return the **DropLines** object. The following example turns on drop lines for chart group one in embedded chart one and then sets the drop line color to red.

```
Worksheets("sheet1").ChartObjects(1).Activate
ActiveChart.ChartGroups(1).HasDropLines = True
ActiveChart.ChartGroups(1).DropLines.Border.ColorIndex = 3
```

Remarks

If the **HasDropLines** property is **False**, most properties of the **DropLines** object are disabled.

ErrorBars Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjErrorBarsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjErrorBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjErrorBarsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjErrorBarsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjErrorBarsE"}

Represents the error bars on a chart series. Error bars indicate the degree of uncertainty for chart data. Only series in area, bar, column, line, and scatter groups on a 2-D chart can have error bars. Only series in scatter groups can have x and y error bars. This object isn't a collection. There's no object that represents a single error bar; you either have x error bars or y error bars turned on for all points in a series or you have them turned off.

Using the ErrorBars Object

Use the **ErrorBars** property to return the **ErrorBars** object. The following example turns on error bars for series one in embedded chart one and then sets the end style for the error bars.

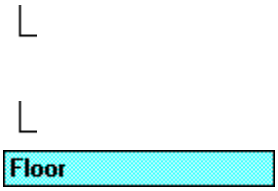
```
Worksheets("sheet1").ChartObjects(1).Activate
ActiveChart.SeriesCollection(1).HasErrorBars = True
ActiveChart.SeriesCollection(1).ErrorBars.EndStyle = xlNoCap
```

Remarks

The **ErrorBar** method changes the error bar format and type.

Floor Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjFloorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjFloorX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjFloorP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjFloorM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjFloorE"}



Represents the floor of a 3-D chart

Using the Floor Object

Use the **Floor** property to return the **Floor** object. The following example sets the floor color for embedded chart one to cyan. The example will fail if the chart isn't a 3-D chart.

```
Worksheets("sheet1").ChartObjects(1).Activate
ActiveChart.Floor.Interior.Color = RGB(0, 255, 255)
```

Font Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjFontC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjFontP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjFontM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjFontE"}

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

Using the Font Object

Use the **Font** property to return the **Font** object. The following example formats cells A1:C5 as bold.

```
Worksheets("sheet1").Range("a1:c5").Font.Bold = True
```

If you don't want to format all the text in a cell or graphic the same way, use the **Characters** property to return a subset of the text.

AxisTitle, **Characters**, **ChartArea**, **ChartTitle**, **DataLabel**, **DataLabels**, **Legend**, **LegendEntry**, **Range**, **Style**, **TickLabels**

Gridlines Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjGridlinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjGridlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjGridlinesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjGridlinesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjGridlinesE"}

Represents major or minor gridlines on a chart axis. Gridlines extend the tick marks on a chart axis to make it easier to see the values associated with the data markers. This object isn't a collection. There's no object that represents a single gridline; you either have all gridlines for an axis turned on or all of them turned off.

Using the Gridlines Object

Use the **MajorGridlines** property to return the **GridLines** object that represents the major gridlines for the axis. Use the **MinorGridlines** property to return the **GridLines** object that represents the minor gridlines. It's possible to return both major and minor gridlines at the same time.

The following example turns on major gridlines for the category axis on the chart sheet named "Chart1" and then formats the gridlines to be blue dashed lines.

```
With Charts("chart1").Axes(xlCategory)
   .HasMajorGridlines = True
   .MajorGridlines.Border.Color = RGB(0, 0, 255)
   .MajorGridlines.Border.LineStyle = xlDash
End With
```

HiLoLines Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjHiLoLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjHiLoLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjHiLoLinesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjHiLoLinesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjHiLoLinesE"}

Represents the high-low lines in a chart group. High-low lines connect the highest point with the lowest point in every category in the chart group. Only 2-D line groups can have high-low lines. This object isn't a collection. There's no object that represents a single high-low line; you either have high-low lines turned on for all points in a chart group or you have them turned off.

Using the HiLoLines Object

Use the **HiLoLines** property to return the **HiLoLines** object. The following example uses the **AutoFormat** method to create a high-low-close stock chart in embedded chart one (the chart must contain three series) on worksheet one. The example then makes the high-low lines blue.

```
Worksheets(1).ChartObjects(1).Activate
ActiveChart.AutoFormat gallery:=xlLine, format:=8
ActiveChart.ChartGroups(1).HiLoLines.Border.Color = RGB(0, 0, 255)
```

Remarks

If the **HasHiLoLines** property is **False**, most properties of the **HiLoLines** object are disabled.

Interior Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjInteriorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjInteriorX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjInteriorP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjInteriorM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjInteriorE"}

Represents the interior of an object.

Using the Interior Object

Use the **Interior** property to return the **Interior** object. The following example sets the color for the interior of cell A1 to red.

```
Worksheets("sheet1").Range("a1").Interior.ColorIndex = 3
```

AxisTitle, **ChartArea**, **ChartObject**, **ChartObjects**, **ChartTitle**, **DataLabel**, **DataLabels**, **DownBars**, **FormatCondition**, **Floor**, **Legend**, **LegendKey**, **OLEObject**, **OLEObjects**, **PlotArea**, **Point**, **Range**, **Series**, **Style**, **UpBars**, **Walls**

Legend Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjLegendC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjLegendX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjLegendP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjLegendM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjLegendE"}

Represents the legend in a chart. Each chart can have only one legend.The **Legend** object contains one or more **LegendEntry** objects; each **LegendEntry** object contains a **LegendKey** object.

Using the Legend Object

Use the **Legend** property to return the **Legend** object. The following example sets the font style for the legend in embedded chart one on worksheet one to bold.

```
Worksheets(1).ChartObjects(1).Chart.Legend.Font.Bold = True
```

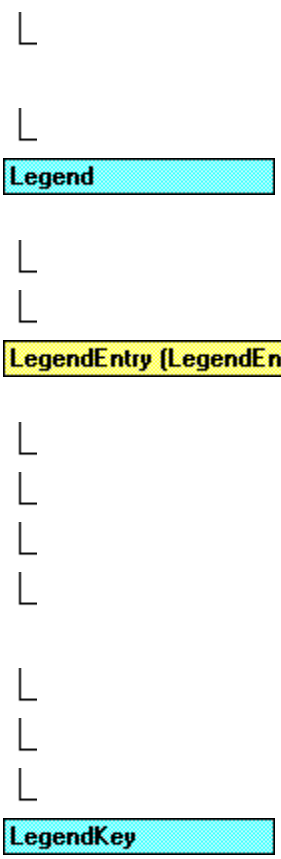
Remarks

The chart legend isn't visible unless the **HasLegend** property is **True**. If this property is **False**,

properties and methods of the **Legend** object will fail.

LegendEntries Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjLegendEntriesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjLegendEntriesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjLegendEntriesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjLegendEntriesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjLegendEntriesE"}



A collection of all the **LegendEntry** objects in the specified chart legend. Each legend entry has two parts: the text of the entry, which is the name of the series or trendline associated with the legend entry; and the entry marker, which visually links the legend entry with its associated series or trendline in the chart. The formatting properties for the entry marker and its associated series or trendline are contained in the **LegendKey** object.

Using the LegendEntries Collection

Use the **LegendEntries** method to return the **LegendEntries** collection. The following example loops through the collection of legend entries in embedded chart one and changes their font color.

```
With Worksheets("sheet1").ChartObjects(1).Chart.Legend
    For i = 1 To .LegendEntries.Count
        .LegendEntries(i).Font.ColorIndex = 5
    Next
End With
```

Use **LegendEntries**(*index*), where *index* is the legend entry index number, to return a single **LegendEntry** object. You cannot return legend entries by name.

The index number represents the position of the legend entry in the legend. `LegendEntries(1)` is at the top of the legend; `LegendEntries(LegendEntries.Count)` is at the bottom. The following example changes the font style for the text of the legend entry at the top of the legend (this is usually the legend for series one) in embedded chart one to italic.

```
Worksheets("sheet1").ChartObjects(1).Chart _
    .Legend.LegendEntries(1).Font.Italic = True
```

LegendEntry Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjLegendEntryC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjLegendEntryX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjLegendEntryP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjLegendEntryM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjLegendEntryE"}

Represents a legend entry in a chart legend. The **LegendEntry** object is a member of the **LegendEntries** collection. The **LegendEntries** collection contains all the **LegendEntry** objects in the legend.

Each legend entry has two parts: the text of the entry, which is the name of the series associated with the legend entry; and an entry marker, which visually links the legend entry with its associated series or trendline in the chart. Formatting properties for the entry marker and its associated series or trendline are contained in the **LegendKey** object.

The text of a legend entry cannot be changed. **LegendEntry** objects support font formatting, and they can be deleted. No pattern formatting is supported for legend entries. The position and size of entries is fixed.

Using the LegendEntry Object

Use **LegendEntries**(*index*), where *index* is the legend entry index number, to return a single **LegendEntry** object. You cannot return legend entries by name.

The index number represents the position of the legend entry in the legend. `LegendEntries(1)` is at the top of the legend, and `LegendEntries(LegendEntries.Count)` is at the bottom. The following example changes the font for the text of the legend entry at the top of the legend (this is usually the legend for series one) in embedded chart one on the worksheet named "Sheet1."

```
Worksheets("sheet1").ChartObjects(1).Chart _
```

```
    .Legend.LegendEntries(1).Font.Italic = True
```

Remarks

There's no direct way to return the series or trendline corresponding to the legend entry.

After legend entries have been deleted, the only way to restore them is to remove and recreate the legend that contained them by setting the **HasLegend** property for the chart to **False** and then back to **True**.

LegendKey Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjLegendKeyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjLegendKeyX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjLegendKeyP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjLegendKeyM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjLegendKeyE"}

Represents a legend key in a chart legend. Each legend key is a graphic that visually links a legend entry with its associated series or trendline in the chart. The legend key is linked to its associated

series or trendline in such a way that changing the formatting of one simultaneously changes the formatting of the other.

Using the LegendKey Object

Use the **LegendKey** property to return the **LegendKey** object. The following example changes the marker background color for the legend entry at the top of the legend for embedded chart one on the worksheet named "Sheet1." This simultaneously changes the format of every point in the series associated with this legend entry. The associated series must support data markers.

```
Worksheets("sheet1").ChartObjects(1).Chart _
    .Legend.LegendEntries(1).LegendKey.MarkerBackgroundColorIndex = 5
```

Mailer Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjMailerC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjMailerX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjMailerP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjMailerM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjMailerE"}

Mailer

Represents the PowerTalk Mailer for a workbook. This object is available only on the Macintosh, with the PowerTalk system extension installed.

Using the Mailer Object

Use the **Mailer** property to return the **Mailer** object. The following example sets the **Subject** property for the mailer attached to the active workbook.

```
ActiveWorkbook.HasMailer = True
ActiveWorkbook.Mailer.Subject = "Here is the workbook."
```

Name Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjNameP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjNameM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjNameE"}

Names (Name)

Name

Represents a defined name for a range of cells. Names can be either built-in names – such as Database, Print_Area, and Auto_Open – or custom names.

Application, Workbook, and Worksheet Objects

The **Name** object is a member of the **Names** collection for the **Application**,**Workbook**, and **Worksheet** objects. Use **Names**(*index*), where *index* is the name index number or defined name, to return a single **Name** object.

The index number indicates the position of the name within the collection. Names are placed in alphabetic order, from a to z, and are not case-sensitive (this is the same order as is displayed in the **Define Name** and **Apply Names** dialog boxes, returned by clicking the **Name** command on the **Insert** menu). The following example displays the cell reference for the first name in the application collection.

```
MsgBox Names(1).RefersTo
```

The following example deletes the name "mySortRange" from the active workbook.

```
ActiveWorkbook.Names("mySortRange").Delete
```

Use the **Name** property to return or set the text of the name itself. The following example changes the name of the first **Name** object in the active workbook.

```
Names(1).Name = "stock_values"
```

Range Objects

Although a **Range** object can have more than one name, there's no **Names** collection for the **Range** object. Use **Name** with a **Range** object to return the first name from the list of names (sorted alphabetically) assigned to the range. The following example sets the **Visible** property for the first name assigned to cells A1:B1 on worksheet one.

```
Worksheets(1).Range("a1:b1").Name.Visible = False
```

Names Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjNamesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjNamesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjNamesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjNamesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjNamesE"}

A collection of all the **Name** objects in the application or workbook. Each **Name** object represents a defined name for a range of cells. Names can be either built-in names – such as Database, Print_Area, and Auto_Open – or custom names.

Using the Names Collection

Use the **Names** property to return the **Names** collection. The following example creates a list of all the

names in the active workbook, plus the addresses they refer to.

```
Set nms = ActiveWorkbook.Names
Set wks = Worksheets(1)
For r = 1 To nms.Count
    wks.Cells(r, 2).Value = nms(r).Name
    wks.Cells(r, 3).Value = nms(r).RefersToRange.Address
Next
```

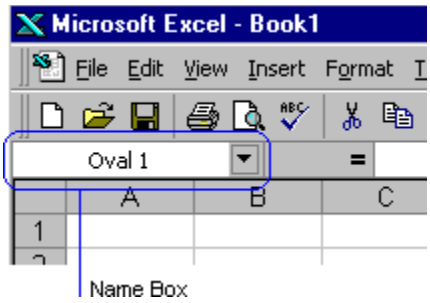
Use the **Add** method to create a name and add it to the collection.The following example creates a new name that refers to cells A1:C20 on the worksheet named "Sheet1."

```
Names.Add Name:="test", RefersTo:="=sheet1!$a$1:$c$20"
```

The ***RefersTo*** argument must be specified in A1-style notation, including dollar signs ($) where appropriate. For example, if cell A10 is selected on Sheet1 and you define a name by using the ***RefersTo*** argument "=sheet1!A1:B1", the new name actually refers to cells A10:B10 (because you specified a relative reference). To specify an absolute reference, use "=sheet1!$A$1:$B$1".

Use **Names**(*index*), where *index* is the name index number or defined name, to return a single **Name** object.The following example deletes the name "mySortRange" from the active workbook.

```
ActiveWorkbook.Names("mySortRange").Delete
```


Microsoft Excel - Book1
File Edit View Insert Format T
Oval 1
=
A
B
C
1
Name Box

Outline Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjOutlineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjOutlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjOutlineP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjOutlineM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjOutlineE"}

Outline

Represents an outline on a worksheet.

Using the Outline Object

Use the **Outline** property to return an **Outline** object. The following example sets the outline on Sheet4 so that only the first outline level is shown.

```
Worksheets("sheet4").Outline.ShowLevels 1
```

PageSetup Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPageSetupC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPageSetupX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPageSetupP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPageSetupM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPageSetupE"}

PageSetup

Represents the page setup description. The **PageSetup** object contains all page setup attributes (left margin, bottom margin, paper size, and so on) as properties.

Using the PageSetup Object

Use the **PageSetup** property to return a **PageSetup** object. The following example sets the orientation to landscape mode and then prints the worksheet.

```
With Worksheets("sheet1")
    .PageSetup.Orientation = xlLandscape
    .PrintOut
End With
```

The **With** statement makes it easier and faster to set several properties at the same time. The following example sets all the margins for worksheet one.

```
With Worksheets(1).PageSetup
     .LeftMargin = Application.InchesToPoints(0.5)
     .RightMargin = Application.InchesToPoints(0.75)
     .TopMargin = Application.InchesToPoints(1.5)
     .BottomMargin = Application.InchesToPoints(1)
     .HeaderMargin = Application.InchesToPoints(0.5)
     .FooterMargin = Application.InchesToPoints(0.5)
End With
```

Chart, **Worksheet**

Pane Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPaneC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPaneX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPaneP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPaneM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPaneE"}

Windows (Window)

└

└

Panes (Pane)

Represents a pane of a window. **Pane** objects exist only for worksheets and Microsoft Excel 4.0 macro sheets. The **Pane** object is a member of the **Panes** collection. The **Panes** collection contains all of the panes shown in a single window.

Using the Pane Object

Use **Panes**(*index*), where *index* is the pane index number, to return a single **Pane** object. The following example splits the window in which worksheet one is displayed and then scrolls through the pane in the lower-left corner until row five is at the top of the pane.

```
Worksheets(1).Activate
ActiveWindow.Split = True
ActiveWindow.Panes(3).ScrollRow = 5
```

Panes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPanesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPanesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPanesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPanesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPanesE"}

Windows (Window)

└

Panes (Pane)

A collection of all the **Pane** objects shown in the specified window. **Pane** objects exist only for worksheets and Microsoft Excel 4.0 macro sheets.

Using the Panes Collection

Use the **Panes** property to return the **Panes** collection. The following example freezes panes in the active window if the window contains more than one pane.

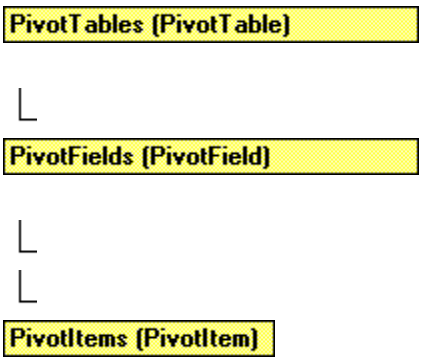
```
If ActiveWindow.Panes.Count > 1 Then ActiveWindow.FreezePanes = True
```

Use **Panes**(*index*), where *index* is the pane index number, to return a single **Pane** object. The following example scrolls through the upper-left pane of the window in which Sheet1 is displayed.

```
Worksheets("sheet1").Activate
Windows(1).Panes(1).LargeScroll down:=1
```

PivotField Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotFieldC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPivotFieldX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotFieldP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotFieldM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotFieldE"}



Represents a field in a PivotTable. The **PivotField** object is a member of the **PivotFields** collection. The **PivotFields** collection contains all the fields objects in a PivotTable, including hidden fields.

Using the PivotField Object

Use **PivotFields**(*index*), where *index* is the pivot-field name or index number, to return a single **PivotField** object. The following example makes the Year field a row field in PivotTable one on Sheet3.

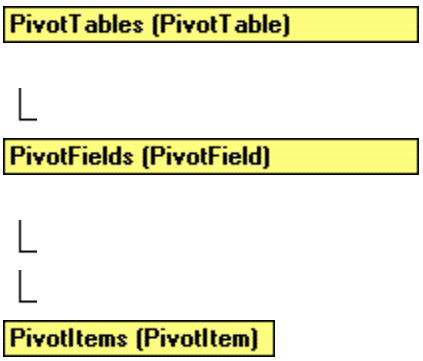
```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields("year").Orientation = xlRowField
```

In some cases, it may be easier to use one of the properties that returns a subset of the PivotTable fields. The following properties are available:

- **ColumnFields** property
- **DataFields** property
- **HiddenFields** property
- **PageFields** property
- **RowFields** property
- **VisibleFields** property

PivotFields Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotFieldsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPivotFieldsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotFieldsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotFieldsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotFieldsE"}



A collection of all the **PivotField** objects in a PivotTable.

Using the PivotFields Collection

Use the **PivotFields** method to return the **PivotFields** collection. The following example enumerates the pivot field names in PivotTable one on Sheet3.

```
With Worksheets("sheet3").PivotTables(1)
    For i = 1 To .PivotFields.Count
        MsgBox .PivotFields(i).Name
    Next
End With
```

Use **PivotFields**(*index*), where *index* is the pivot-field name or index number, to return a single **PivotField** object. The following example makes the Year field a row field in PivotTable one on Sheet3.

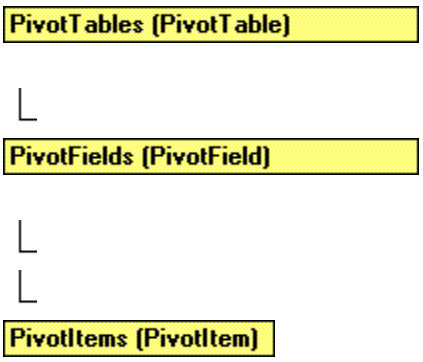
```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields("year").Orientation = xlRowField
```

In some cases, it may be easier to use one of the properties that returns a subset of the PivotTable fields. The following accessor methods are available:

- **ColumnFields** property
- **DataFields** property
- **HiddenFields** property
- **PageFields** property
- **RowFields** property
- **VisibleFields** property

PivotItem Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotItemC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPivotItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotItemP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotItemM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotItemE"}



Represents an item in a pivot field. The items are the individual data entries in a field category. The **PivotItem** object is a member of the **PivotItems** collection. The **PivotItems** collection contains all the items in a **PivotField** object.

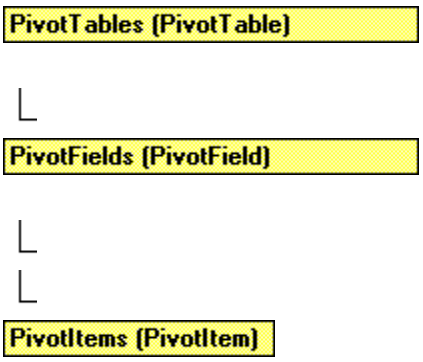
Using the PivotItem Object

Use **PivotItems**(*index*), where *index* is the pivot item index number or name, to return a single **PivotItem** object. The following example hides all entries in PivotTable one on Sheet3 that contain "1998" in the Year field.

```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields("year").PivotItems("1998").Visible = False
```

PivotItems Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPivotItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotItemsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotItemsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotItemsE"}



A collection of all the **PivotItem** objects in a pivot field. The items are the individual data entries in a field category.

Using the PivotItems Collection

Use the **PivotItems** method to return the **PivotItems** collection. The following example creates an enumerated list of pivot field names and the items contained in those fields for PivotTable one on Sheet4.

```
Worksheets("sheet4").Activate
With Worksheets("sheet3").PivotTables(1)
    c = 1
    For i = 1 To .PivotFields.Count
        r = 1
        Cells(r, c) = .PivotFields(i).Name
        r = r + 1
        For x = 1 To .PivotFields(i).PivotItems.Count
            Cells(r, c) = .PivotFields(i).PivotItems(x).Name
            r = r + 1
        Next
        c = c + 1
    Next
End With
```

Use **PivotItems**(*index*), where *index* is the pivot item index number or name to return a single **PivotItem** object. The following example hides all entries in PivotTable one on Sheet3 that contain "1998" in the Year field.

```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields("year").PivotItems("1998").Visible = False
```

PivotTable Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotTableC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPivotTableX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotTableP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotTableM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotTableE"}

PivotTables (PivotTable)

PivotFields (PivotField)

Represents a PivotTable on a worksheet. The **PivotTable** object is a member of the **PivotTables** collection. The **PivotTables** collection contains all the **PivotTable** objects on a single worksheet.

Using the PivotTable Object

Use **PivotTables**(*index*), where *index* is the PivotTable index number or name, to return a single **PivotTable** object. The following example makes the field named "year" a row field in PivotTable one on Sheet3.

```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields("year").Orientation = xlRowField
```

Remarks

Because PivotTable programming can be complex, it's generally easiest to record PivotTable actions and then revise the recorded code. To record a macro, point to **Macro** on the **Tools** menu and click **Record New Macro**.

PivotTables Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotTablesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPivotTablesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotTablesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotTablesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotTablesE"}

PivotFields (PivotField)

A collection of all the **PivotTable** objects on the specified worksheet.

Using the PivotTables Collection

Use the **PivotTables** method to return the **PivotTables** collection. The following example displays the number of PivotTables on Sheet3.

```
MsgBox Worksheets("sheet3").PivotTables.Count
```

Use the **PivotTableWizard** method to create a new PivotTable and add it to the collection. The following example creates a new PivotTable from a Microsoft Excel database (contained in the range A1:C100).

```
ActiveSheet.PivotTableWizard xlDatabase, Range("A1:C100")
```

Use **PivotTables**(*index*), where *index* is the PivotTable index number or name, to return a single **PivotTable** object. The following example makes the Year field a row field in PivotTable one on Sheet3.

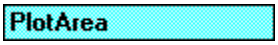
```
Worksheets("sheet3").PivotTables(1) _
    .PivotFields("year").Orientation = xlRowField
```

Remarks

Because PivotTable programming can be complex, it's generally easiest to record PivotTable actions and then revise the recorded code. To record a macro, point to **Macro** on the **Tools** menu and click **Record New Macro**.

PlotArea Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPlotAreaC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPlotAreaX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPlotAreaP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPlotAreaM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPlotAreaE"}



Represents the plot area of a chart. This is the area where your chart data is plotted. The plot area on a 2-D chart contains the data markers, gridlines, data labels, trendlines, and optional chart items placed in the chart area. The plot area on a 3-D chart contains all the above items plus the walls, floor, axes, axis titles, and tick-mark labels in the chart.

The plot area is surrounded by the chart area. The chart area on a 2-D chart contains the axes, the chart title, the axis titles, and the legend. The chart area on a 3-D chart contains the chart title and the legend. For information about formatting the chart area, see the **ChartArea** object.

Using the PlotArea Object

Use the **PlotArea** property to return a **PlotArea** object. The following example activates the chart sheet named "Chart1," places a dashed border around the chart area of the active chart, and places a dotted border around the plot area.

```
Charts("Chart1").Activate
With ActiveChart
    .ChartArea.Border.LineStyle = xlDash
    .PlotArea.Border.LineStyle = xlDot
End With
```

Point Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPointC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPointX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPointP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPointM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPointE"}

Represents a single point in a series in a chart. The **Point** object is a member of the **Points** collection. The **Points** collection contains all the points in one series.

Using the Point Object

Use **Points**(*index*), where *index* is the point index number, to return a single **Point** object. Points are numbered from left to right on the series. `Points(1)` is the leftmost point, and

`Points(Points.Count)` is the rightmost point. The following example sets the marker style for the third point in series one in embedded chart one on worksheet one. The specified series must be a 2-D line, scatter, or radar series.

```
Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection(1).Points(3).MarkerStyle = xlDiamond
```

Points Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPointsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjPointsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPointsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPointsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPointsE"}

A collection of all the **Point** objects in the specified series in a chart.

Using the Points Collection

Use the **Points** method to return the **Points** collection. The following example adds a data label to the last point on series one in embedded chart one on worksheet one.

```
Dim pts As Points
```

```
Set pts = Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection(1).Points
pts(pts.Count).ApplyDataLabels type:=xlShowValue
```

Use **Points**(*index*), where *index* is the point index number, to return a single **Point** object. Points are numbered from left to right on the series. `Points(1)` is the leftmost point, and `Points(Points.Count)` is the rightmost point. The following example sets the marker style for the third point in series one in embedded chart one on worksheet one. The specified series must be a 2-D line, scatter, or radar series.

```
Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection(1).Points(3).MarkerStyle = xlDiamond
```

Range Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjRangeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjRangeM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjRangeE"}

Represents a cell, a row, a column, a selection of cells containing one or more contiguous blocks of cells, or a 3-D range.

Using the Range Object

The following properties and methods for returning a **Range** object are described in this section:

- **Range** property
- **Cells** property
- **Range** and **Cells**
- **Offset** property
- **Union** method

Range Property

Use **Range**(*arg*), where *arg* names the range, to return a **Range** object that represents a single cell or a range of cells. The following example places the value of cell A1 in cell A5.

```
Worksheets("Sheet1").Range("A5").Value = _
    Worksheets("Sheet1").Range("A1").Value
```

The following example fills the range A1:H8 with random numbers by setting the formula for each cell in the range. When it's used without an object qualifier (an object to the left of the period), the **Range** property returns a range on the active sheet. If the active sheet isn't a worksheet, the method fails. Use the **Activate** method to activate a worksheet before you use the **Range** property without an explicit object qualifier.

```
Worksheets("sheet1").Activate
Range("A1:H8").Formula = "=rand()" 'Range is on the active sheet
```

The following example clears the contents of the range named "Criteria."

```
Worksheets(1).Range("criteria").ClearContents
```

If you use a text argument for the range address, you must specify the address in A1-style notation (you cannot use R1C1-style notation).

Cells Property

Use **Cells**(*row*, *column*) where *row* is the row index and *column* is the column index, to return a single cell. The following example sets the value of cell A1 to 24.

```
Worksheets(1).Cells(1, 1).Value = 24
```

The following example sets the formula for cell A2.

```
ActiveSheet.Cells(2, 1).Formula = "=sum(b1:b5)"
```

Although you can also use `Range("A1")` to return cell A1, there may be times when the **Cells** property is more convenient because you can use a variable for the row or column. The following example creates column and row headings on Sheet1. Notice that after the worksheet has been activated, the **Cells** property can be used without an explicit sheet declaration (it returns a cell on the active sheet).

```
Sub SetUpTable()
Worksheets("sheet1").Activate
For theYear = 1 To 5
   Cells(1, theYear + 1).Value = 1990 + theYear
Next theYear
'
For theQuarter = 1 To 4
   Cells(theQuarter + 1, 1).Value = "Q" & theQuarter
Next theQuarter
End Sub
```

Although you could use Visual Basic string functions to alter A1-style references, it's much easier (and much better programming practice) to use the `Cells(1, 1)` notation.

Use *expression*.**Cells**(*row*, *column*) , where *expression* is an expression that returns a **Range** object, and *row* and *column* are relative to the upper-left corner of the range, to return part of a range. The following example sets the formula for cell C5.

```
Worksheets(1).Range("c5:c10").Cells(1, 1).Formula = "=rand()"
```

Range and Cells

Use **Range**(*cell1, cell2*), where *cell1* and *cell2* are **Range** objects that specify the start and end cells, to return a **Range** object. The following example sets the border line style for cells 1:J10.

```
With Worksheets(1)
    .Range(.Cells(1, 1), .Cells(10, 10)).Borders.LineStyle = xlThick
End With
```

Notice the period in front of each occurrence of the **Cells** property. The period is required if the result of the preceding **With** statement is to be applied to the **Cells** property – in this case, to indicate that the cells are on worksheet one (without the period, the **Cells** property would return cells on the active sheet).

Offset Property

Use **Offset**(*row, colu*mn), where *row* and *column* are the row and column offsets, to return a range at a specified offset to another range. The following example selects the cell three rows down from and one column to the right of the cell in the upper-left corner of the current selection. You cannot select a cell that isn't on the active sheet, so you must first activate the worksheet.

```
Worksheets("sheet1").Activate  'can't select unless the sheet is active
Selection.Offset(3, 1).Range("a1").Select
```

Union Method

Use **Union**(*range1, range2*, ...) to return multiple-area ranges – that is, ranges composed of two or more contiguous blocks of cells. The following example creates an object defined as the union of ranges A1:B2 and C3:D4, and then selects the defined range.

```
Dim r1 As Range, r2 As Range, myMultiAreaRange As Range
Worksheets("sheet1").Activate
Set r1 = Range("A1:B2")
Set r2 = Range("C3:D4")
Set myMultiAreaRange = Union(r1, r2)
myMultiAreaRange.Select
```

If you work with selections that contain more than one area, the **Areas** property is very useful. It divides a multiple-area selection into individual **Range** objects and then returns the objects as a collection. You can use the **Count** property on the returned collection to check for a selection that contains more than one area, as shown in the following example.

```
Sub NoMultiAreaSelection()
   numberOfSelectedAreas = Selection.Areas.Count
   If numberOfSelectedAreas > 1 Then
      MsgBox "You cannot carry out this command " & _
         "on multi-area selections"
   End If
End Sub
```

Areas, **Borders**, **Characters**, **Font**, **Interior**, **Name**, **Style**

RoutingSlip Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjRoutingSlipC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjRoutingSlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjRoutingSlipP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjRoutingSlipM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjRoutingSlipE"}

RoutingSlip

Represents the routing slip for a workbook. The routing slip is used to send a workbook through the electronic mail system.

Using the RoutingSlip Object

Use the **RoutingSlip** property to return the **RoutingSlip** object. The following example sets the delivery style for the routing slip attached to the active workbook. For a more detailed example, see the **RoutingSlip** property.

```
ActiveWorkbook.HasRoutingSlip = True
ActiveWorkbook.RoutingSlip.Delivery = xlOneAfterAnother
```

Remarks

The **RoutingSlip** object doesn't exist and cannot be returned unless the **HasRoutingSlip** property for the workbook is **True**.

Scenario Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjScenarioC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjScenarioX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjScenarioP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjScenarioM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjScenarioE"}

Scenarios (Scenario)

Represents a scenario on a worksheet. A scenario is a group of input values (called *changing cells*) that's named and saved. The **Scenario** object is a member of the **Scenarios** collection. The **Scenarios** collection contains all the defined scenarios for a worksheet.

Using the Scenario Object

Use **Scenarios**(*index*), where *index* is the scenario name or index number, to return a single **Scenario** object. The following example shows the scenario named "Typical" on the worksheet named "Options."

```
Worksheets("options").Scenarios("typical").Show
```

Scenarios Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjScenariosC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjScenariosX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjScenariosP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjScenariosM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjScenariosE"}

Scenarios (Scenario)

A collection of all the **Scenario** objects on the specified worksheet. A scenario is a group of input values (called *changing cells*) that's named and saved.

Using the Scenarios Collection

Use the **Scenarios** method to return the **Scenarios** collection. The following example creates a summary for the scenarios on the worksheet named "Options," using cells J10 and J20 as the result cells.

```
Worksheets("options").Scenarios.CreateSummary _
    resultCells:=Worksheets("options").Range("j10,j20")
```

Use the **Add** method to create a new scenario and add it to the collection. The following example adds a new scenario named "Typical" to the worksheet named "Options." The new scenario has two changing cells, A2 and A12, with the respective values 55 and 60.

```
Worksheets("options").Scenarios.Add name:="Typical", _
    changingCells:=Worksheets("options").Range("A2,A12"), _
    values:=Array("55", "60")
```

Use **Scenarios**(*index*), where *index* is the scenario name or index number, to return a single **Scenario** object. The following example shows the scenario named "Typical" on the worksheet named "Options."

```
Worksheets("options").Scenarios("typical").Show
```

Series Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjSeriesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjSeriesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjSeriesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjSeriesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjSeriesE"}

Represents a series in a chart. The **Series** object is a member of the **SeriesCollection** collection.

Using the Series Object

Use **SeriesCollection**(*index*), where *index* is the series index number or name, to return a single **Series** object. The following example sets the color of the interior for the first series in embedded chart one on Sheet1.

```
Worksheets("sheet1").ChartObjects(1).Chart. _
    SeriesCollection(1).Interior.Color = RGB(255, 0, 0)
```

The series index number indicates the order in which the series were added to the chart. `SeriesCollection(1)` is the first series added to the chart, and `SeriesCollection(SeriesCollection.Count)` is the last one added.

SeriesCollection Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjSeriesCollectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjSeriesCollectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjSeriesCollectionP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjSeriesCollectionM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjSeriesCollectionE"}

A collection of all the **Series** objects in the specified chart or chart group.

Using the SeriesCollection Collection

Use the **SeriesCollection** method to return the **SeriesCollection** collection. The following example adds the data in cells C1:C10 on worksheet one to an existing series in the series collection in embedded chart one.

```
Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection.Extend Worksheets(1).Range("c1:c10")
```

Use the **Add** method to create a new series and add it to the chart. The following example adds the data from cells A1:A19 as a new series on the chart sheet named "Chart1."

```
Charts("chart1").SeriesCollection.Add _
    source:=Worksheets("sheet1").Range("a1:a19")
```

Use **SeriesCollection**(*index*), where *index* is the series index number or name, to return a single **Series** object. The following example sets the color of the interior for the first series in embedded chart one on Sheet1.

```
Worksheets("sheet1").ChartObjects(1).Chart. _
    SeriesCollection(1).Interior.Color = RGB(255, 0, 0)
```

SeriesLines Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjSeriesLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjSeriesLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjSeriesLinesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjSeriesLinesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjSeriesLinesE"}

Represents series lines in a chart group. Series lines connect the data values from each series. Only 2-D stacked bar or column chart groups can have series lines. This object isn't a collection. There's no object that represents a single series line; you either have series lines turned on for all points in a chart group or you have them turned off.

Using the SeriesLines Object

Use the **SeriesLines** property to return a **SeriesLines** object. The following example adds series lines to chart group one in embedded chart one on worksheet one (the chart must be a 2-D stacked bar or column chart).

```
With Worksheets(1).ChartObjects(1).Chart.ChartGroups(1)
    .HasSeriesLines = True
    .SeriesLines.Border.Color = RGB(0, 0, 255)
End With
```

Remarks

If the **HasSeriesLines** property is **False**, most properties of the **SeriesLines** object are disabled.

Sheets Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjSheetsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjSheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjSheetsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjSheetsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjSheetsE"}

Sheets

A collection of all the sheets in the specified or active workbook. The **Sheets** collection can contain **Chart** or **Worksheet** objects.

The **Sheets** collection is useful when you want to return sheets of any type. If you need to work with sheets of only one type, see the object topic for that sheet type.

Using the Sheets Collection

Use the **Sheets** property to return the **Sheets** collection. The following example prints all sheets in the active workbook.

```
Sheets.PrintOut
```

Use the **Add** method to create a new sheet and add it to the collection. The following example adds two chart sheets to the active workbook, placing them after sheet two in the workbook.

```
Sheets.Add type:=xlChart, count:=2, after:=Sheets(2)
```

Use **Sheets**(*index*), where *index* is the sheet name or index number, to return a single **Chart** or **Worksheet** object. The following example activates the sheet named "sheet1."

```
Sheets("sheet1").Activate
```

Use **Sheets**(*array*) to specify more than one sheet. The following example moves the sheets named "Sheet4" and "Sheet5" to the beginning of the workbook.

```
Sheets(Array("Sheet4", "Sheet5")).Move before:=Sheets(1)
```

SoundNote Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjSoundNoteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjSoundNoteX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjSoundNoteP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjSoundNoteM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjSoundNoteE"}

This object should not be used. Sound notes have been removed from Microsoft Excel.

Style Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjStyleP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjStyleM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjStyleE"}

Styles (Style)

Borders (Border)

Represents a style description for a range. The **Style** object contains all style attributes (font, number format, alignment, and so on) as properties. There are several built-in styles, including Normal, Currency, and Percent. Using the **Style** object is a fast and efficient way to change several cell-formatting properties on multiple cells at the same time.

For the **Workbook** object, the **Style** object is a member of the **Styles** collection. The **Styles** collection contains all the defined styles for the workbook.

Using the Style Object

Use the **Style** property to return the **Style** object used with a **Range** object. The following example applies the Percent style to cells A1:A10 on Sheet1.

```
Worksheets("sheet1").Range("a1:a10").Style.Name = "percent"
```

You can change the appearance of a cell by changing properties of the style applied to that cell. Keep in mind, however, that changing a style property will affect all cells already formatted with that style.

Use **Styles**(*index*), where *index* is the style index number or name, to return a single **Style** object from the workbook **Styles** collection. The following example changes the Normal style for the active workbook by setting the style's **Bold** property.

```
ActiveWorkbook.Styles("Normal").Font.Bold = True
```

Styles are sorted alphabetically by style name. The style index number denotes the position of the specified style in the sorted list of style names. `Styles(1)` is the first style in the alphabetic list, and `Styles(Styles.Count)` is the last one in the list.

For more information about creating and modifying a style, see the **Styles** object.

Range, **Workbook**

Styles Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjStylesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjStylesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjStylesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjStylesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjStylesE"}

Styles (Style)

A collection of all the **Style** objects in the specified or active workbook. Each **Style** object represents a style description for a range. The **Style** object contains all style attributes (font, number format, alignment, and so on) as properties. There are several built-in styles – including Normal, Currency, and Percent – which are listed in the **Style name** box inthe **Style** dialog box (**Format** menu).

Using the Styles Collection

Use the **Styles** property to return the **Styles** collection. The following example creates a list of style names on worksheet one in the active workbook.

```
For i = 1 To ActiveWorkbook.Styles.Count
    Worksheets(1).Cells(i, 1) = ActiveWorkbook.Styles(i).Name
Next
```

Use the **Add** method to create a new style and add it to the collection. The following example creates a new style based on the Normal style, modifies the border and font, and then applies the new style to cells A25:A30.

```
With ActiveWorkbook.Styles.Add(name:="bookman top border")
   .Borders(xlTop).LineStyle = xlDouble
   .Font.Bold = True
   .Font.Name = "bookman"
End With
Worksheets(1).Range("a25:a30").Style = "bookman top border"
```

Use **Styles**(*index*), where *index* is the style index number or name, to return a single **Style** object from the workbook **Styles** collection. The following example changes the Normal style for the active

workbook by setting its **Bold** property.

```
ActiveWorkbook.Styles("Normal").Font.Bold = True
```

TickLabels Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjTickLabelsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjTickLabelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjTickLabelsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjTickLabelsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjTickLabelsE"}

Represents the tick-mark labels associated with tick marks on a chart axis. This object isn't a collection. There's no object that represents a single tick-mark label; you must return all the tick-mark labels as a unit.

Tick-mark label text for the category axis comes from the name of the associated category in the chart. The default tick-mark label text for the category axis is the number that indicates the position of the category relative to the left end of this axis. To change the number of unlabeled tick marks between tick-mark labels, you must change the **TickLabelSpacing** property for the category axis.

Tick-mark label text for the value axis is calculated based on the **MajorUnit**, **MinimumScale**, and **MaximumScale** properties of the value axis. To change the tick-mark label text for the value axis, you must change thte values of these properties.

Using the TickLabels Object

Use the **TickLabels** property to return the **TickLabels** object. The following example sets the number format for the tick-mark labels on the value axis in embedded chart one on Sheet1.

```
Worksheets("sheet1").ChartObjects(1).Chart _
    .Axes(xlValue).TickLabels.NumberFormat = "0.00"
```

Trendline Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjTrendlineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjTrendlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjTrendlineP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjTrendlineM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjTrendlineE"}

seriesco.bmp}

Represents a trendline in a chart. A trendline shows the trend, or direction, of data in a series. The **Trendline** object is a member of the **Trendlines** collection. The **Trendlines** collection contains all the **Trendline** objects for a single series.

Using the Trendline Object

Use **Trendlines**(*index*), where *index* is the trendline index number, to return a single **Trendline** object. The following example changes the trendline type for the first series in embedded chart one on worksheet one. If the series has no trendline, this example will fail.

```
Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection(1).Trendlines(1).Type = xlMovingAvg
```

The index number denotes the order in which the trendlines were added to the series. `Trendlines(1)` is the first trendline added to the series, and `Trendlines(Trendlines.Count)` is the last one added.

Trendlines Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjTrendlinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjTrendlinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjTrendlinesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjTrendlinesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjTrendlinesE"}

seriesco.bmp}

A collection of all the **Trendline** objects for the specified series. Each **Trendline** object represents a trendline in a chart. A trendline shows the trend, or direction, of data in a series.

Using the Trendlines Collection

Use the **Trendlines** method to return the **Trendlines** collection. The following example displays the number of trendlines for series one in Chart1.

```
MsgBox Charts(1).SeriesCollection(1).Trendlines.Count
```

Use the **Add** method to create a new trendline and add it to the series. The following example adds a linear trendline to the first series in embedded chart one on Sheet1.

```
Worksheets("sheet1").ChartObjects(1).Chart.SeriesCollection(1) _
    .Trendlines.Add type:=xlLinear, name:="Linear Trend"
```

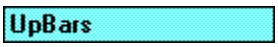
Use **Trendlines**(*index*), where *index* is the trendline index number, to return a single **TrendLine** object. The following example changes the trendline type for the first series in embedded chart one on worksheet one. If the series has no trendline, this example will fail.

```
Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection(1).Trendlines(1).Type = xlMovingAvg
```

The index number denotes the order in which the trendlines were added to the series. `Trendlines(1)` is the first trendline added to the series, and `Trendlines(Trendlines.Count)` is the last one added.

UpBars Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjUpBarsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjUpBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjUpBarsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjUpBarsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjUpBarsE"}



Represents the up bars in a chart group. Up bars connect points on series one with higher values on the last series in the chart group (the lines go up from series one). Only 2-D line groups that contain at least two series can have up bars. This object isn't a collection. There's no object that represents a single up bar; you either have up bars turned on for all points in a chart group or you have them turned off.

Using the UpBars Object

Use the **UpBars** property to return the **UpBars** object. The following example turns on up and down bars for chart group one in embedded chart one on Sheet5. The example then sets the up bar color to blue and sets the down bar color to red.

```
With Worksheets("sheet5").ChartObjects(1).Chart.ChartGroups(1)
    .HasUpDownBars = True
    .UpBars.Interior.Color = RGB(0, 0, 255)
    .DownBars.Interior.Color = RGB(255, 0, 0)
End With
```

Remarks

If the **HasUpDownBars** property is **False**, most properties of the **UpBars** object are disabled.

Walls Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWallsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjWallsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWallsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWallsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWallsE"}

Walls

Represents the walls of a 3-D chart. This object isn't a collection. There's no object that represents a single wall; you must return all the walls as a unit.

Using the Walls Object

Use the **Walls** property to return the **Walls** object. The following example sets the pattern on the walls for embedded chart one on Sheet1. If the chart isn't a 3-D chart, this example will fail.

```
Worksheets("Sheet1").ChartObjects(1).Chart _
    .Walls.Interior.Pattern = xlGray75
```

Window Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWindowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWindowP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWindowM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWindowE"}

Represents a window. Many worksheet characteristics, such as scroll bars and gridlines, are actually properties of the window. The **Window** object is a member of the **Windows** collection. The **Windows** collection for the **Application** object contains all the windows in the application, whereas the **Windows** collection for the **Workbook** object contains only the windows in the specified workbook.

Using the Window Object

Use **Windows**(*index*), where *index* is the window name or index number, to return a single **Window** object. The following example maximizes the active window.

```
Windows(1).WindowState = xlMaximized
```

Note that the active window is always `Windows(1)`.

The window caption is the text shown in the title bar at the top of the window when the window isn't maximized. The caption is also shown in the list of open files on the bottom of the **Windows** menu. Use the **Caption** property to set or return the window caption. Changing the window caption doesn't change the name of the workbook. The following example turns off cell gridlines for the worksheet shown in the Book1.xls:1 window.

```
Windows("book1.xls":1).DisplayGridlines = False
```

Windows Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWindowsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjWindowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWindowsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWindowsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWindowsE"}

A collection of all the **Window** objects in Microsoft Excel. The **Windows** collection for the **Application** object contains all the windows in the application, whereas the **Windows** collection for the **Workbook** object contains only the windows in the specified workbook.

Using the Windows Collection

Use the **Windows** property to return the **Windows** collection. The following example cascades all the windows that are currently displayed in Microsoft Excel.

```
Windows.Arrange arrangeStyle:=xlCascade
```

Use the **NewWindow** method to create a new window and add it to the collection. The following example creates a new window for the active workbook.

```
ActiveWorkbook.NewWindow
```

Use **Windows**(*index*), where *index* is the window name or index number, to return a single **Window** object. The following example maximizes the active window.

```
Windows(1).WindowState = xlMaximized
```

Note that the active window is always `Windows(1)`.

Workbook Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWorkbookC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjWorkbookX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWorkbookP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWorkbookM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWorkbookE"}

Represents a Microsoft Excel workbook. The **Workbook** object is a member of the **Workbooks** collection. The **Workbooks** collection contains all the **Workbook** objects currently open in Microsoft Excel.

Using the Workbook Object

The following properties for returning a **Workbook** object are described in this section:

- **Workbooks** property
- **ActiveWorkbook** property
- **ThisWorkbook** property

Workbooks Property

Use **Workbooks**(*index*), where *index* is the workbook name or index number, to return a single **Workbook** object. The following example activates workbook one.

```
Workbooks(1).Activate
```

The index number denotes the order in which the workbooks were opened or created. `Workbooks(1)` is the first workbook created, and `Workbooks(Workbooks.Count)` is the last one created. Activating a workbook doesn't change its index number. All workbooks are included in the index count, even if they're hidden.

The **Name** property returns the workbook name. You cannot set the name by using this property; if you need to change the name, use the **SaveAs** method to save the workbook under a different name. The following example activates Sheet1 in the workbook named "Cogs.xls" (the workbook must already be open in Microsoft Excel).

```
Workbooks("cogs.xls").Worksheets("sheet1").Activate
```

ActiveWorkbook Property

The **ActiveWorkbook** property returns the workbook that's currently active. The following example sets the name of the author for the active workbook.

```
ActiveWorkbook.Author = "Jean Selva"
```

ThisWorkbook Property

The **ThisWorkbook** property returns the workbook where the Visual Basic code is running. In most

cases, this is the same as the active workbook. However, if the Visual Basic code is part of an add-in, the **ThisWorkbook** property won't return the active workbook. In this case, the active workbook is the workbook calling the add-in, whereas the **ThisWorkbook** property returns the add-in workbook.

If you'll be creating an add-in from your Visual Basic code, you should use the **ThisWorkbook** property to qualify any statement that must be run on the workbook you compile into the add-in.

Charts, **Mailer**, **Names**, **PageSetup**, **RoutingSlip**, **Styles**, **Windows**, **Worksheets**

Workbooks Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWorkbooksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjWorkbooksX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWorkbooksP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWorkbooksM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWorkbooksE"}

A collection of all the **Workbook** objects that are currently open in the Microsoft Excel application.

Using the Workbooks Collection

Use the **Workbooks** property to return the **Workbooks** collection. The following example closes all open workbooks.

```
Workbooks.Close
```

Use the **Add** method to create a new, empty workbook and add it to the collection. The following example adds a new, empty workbook to Microsoft Excel.

```
Workbooks.Add
```

Use the **Open** method to open a file. This creates a new workbook for the opened file. The following example opens the file Array.xls as a read-only workbook.

```
Workbooks.Open fileName:="array.xls", readOnly:=True
```

For more information about using a single **Workbook** object, see the **Workbook** object.

Worksheet Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWorksheetC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjWorksheetX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWorksheetP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWorksheetM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWorksheetE"}

Represents a worksheet. The **Worksheet** object is a member of the **Worksheets** collection. The **Worksheets** collection contains all the **Worksheet** objects in a workbook.

Using the Worksheet Object

The following properties for returning a **Worksheet** object are described in this section:

- **Worksheets** property
- **ActiveSheet** property

Worksheets Property

Use **Worksheets**(*index*), where *index* is the worksheet index number or name to return a single **Worksheet** object. The following example hides worksheet one in the active workbook.

```
Worksheets(1).Visible = False
```

The worksheet index number denotes the position of the worksheet on the workbook's tab bar. `Worksheets(1)` is the first (leftmost) worksheet in the workbook, and `Worksheets(Worksheets.Count)` is the last one. All worksheets are included in the index count, even if they're hidden.

The worksheet name is shown on the tab for the worksheet. Use the **Name** property to set or return the worksheet name. The following example protects the scenarios on Sheet1.

```
Worksheets("sheet1").Protect password:="drowssap", scenarios:=True
```

The **Worksheet** object is also a member of the **Sheets** collection. The **Sheets** collection contains all the sheets in the workbook (both chart sheets and worksheets).

ActiveSheet Property

When a worksheet is the active sheet, you can use the **ActiveSheet** property to refer to it. The following example uses the **Activate** method to activate Sheet 1, sets the page orientation to landscape mode, and then prints the worksheet.

```
Worksheets("sheet1").Activate
ActiveSheet.PageSetup.Orientation = xlLandscape
ActiveSheet.PrintOut
```

ChartObjects, **OLEObjects**, **Outline**, **PageSetup**, **PivotTables**, **Range**, **Scenarios**

Worksheets Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWorksheetsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjWorksheetsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWorksheetsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWorksheetsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWorksheetsE"}

A collection of all the **Worksheet** objects in the specified or active workbook. Each **Worksheet** object represents a worksheet.

Using the Worksheets Collection

Use the **Worksheets** property to return the **Worksheets** collection.The following example moves all the worksheets to the end of the workbook.

```
Worksheets.Move after:=Sheets(Sheets.Count)
```

Use the **Add** method to create a new worksheet and add it to the collection. The following example adds two new worksheets before sheet one of the active workbook.

```
Worksheets.Add count:=2, before:=Sheets(1)
```

Use **Worksheets**(*index*), where *index* is the worksheet index number or name to return a single **Worksheet** object. The following example hides worksheet one in the active workbook.

```
Worksheets(1).Visible = False
```

The **Worksheet** object is also a member of the **Sheets** collection. The **Sheets** collection contains all the sheets in the workbook (both chart sheets and worksheets).

CalculatedFields Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCalculatedFieldsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCalculatedFieldsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCalculatedFieldsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCalculatedFieldsE"}

CalculatedFields (PivotField)

A collection of **PivotField** objects that represents all the calculated fields in the specified PivotTable. For example, a PivotTable that contains Revenue and Expense fields could have a calculated field named "Profit" defined as the amount in the Revenue field minus the amount in the Expense field.

Using the CalculatedFields Collection

Use the **CalculatedFields** method to return the **CalculatedFields** collection The following example deletes the calculated fields from PivotTable one.

```
For Each fld in Worksheets(1).PivotTables("Pivot1").CalculatedFields
   fld.Delete
Next
```

Use **CalculatedFields**(*index*), where *index* is specified field's name or index number, to return a single **PivotField** object from the **CalculatedFields** collection.

CalculatedItems Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCalculatedItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCalculatedItemsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCalculatedItemsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCalculatedItemsE"}

CalculatedItems (PivotItem)

A collection of **PivotItem** objects that represent all the calculated items in the specified PivotTable. For example, a PivotTable that contains January, February, and March items could have a calculated item named "FirstQuarter" defined as the sum of the amounts in January, February, and March.

Using the CalculatedItems Collection

Use the **CalculatedItems** method to return the **CalculatedItems** collection The following example creates a list of the calculated items in PivotTable one, along with their formulas.

```
Set pt = Worksheets(1).PivotTables(1)
For Each ci In pt.PivotFields("Sales").CalculatedItems
   r = r + 1
   With Worksheets(2)
      .Cells(r, 1).Value = ci.Name
      .Cells(r, 2).Value = ci.Formula
   End With
Next
```

Use **CalculatedFields**(*index*), where *index* is the name or index number of the field, to return a single **PivotField** object from the **CalculatedFields** collection.

Comment Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCommentC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCommentP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCommentM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCommentE"}

Comments (Comment)

Represents a cell comment. The **Comment** object is a member of the **Comments** collection.

Using the Comment Object

Use the **Comment** property to return a **Comment** object. The following example changes the text in the comment in cell E5.

```
Worksheets(1).Range("E5").Comment.Text "reviewed on " & Date
```

Use **Comments**(*index*), where *index* is the comment number, to return a single comment from the **Comments** collection. The following example hides comment two on worksheet one.

```
Worksheets(1).Comments(2).Visible = False
```

Use the **AddComment** method to add a comment to a range. The following example adds a comment to cell E5 on worksheet one.

```
With Worksheets(1).Range("e5").AddComment
    .Visible = False
    .Text "reviewed on " & Date
End With
```

Comments Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCommentsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCommentsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCommentsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCommentsE"}

Comments (Comment)

A collection of cell comments. Each comment is represented by a **Comment** object.

Using the Comments Collection

Use the **Comments** property to return the **Comments** collection. The following example hides all the comments on worksheet one.

```
Set cmt = Worksheets(1).Comments
For Each c In cmt
   c.Visible = False
Next
```

Use the **AddComment** method to add a comment to a range. The following example adds a comment to cell E5 on worksheet one.

```
With Worksheets(1).Range("e5").AddComment
   .Visible = False
   .Text "reviewed on " & Date
End With
```

Use **Comments**(*index*), where *index* is the comment number, to return a single comment from the **Comments** collection. The following example hides comment two on worksheet one.

```
Worksheets(1).Comments(2).Visible = False
```

CustomView Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCustomViewC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCustomViewP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCustomViewM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCustomViewE"}

CustomViews (CustomView)

Represents a custom workbook view. The **CustomView** object is a member of the **CustomViews** collection.

Using the CustomView Object

Use **CustomViews**(*index*), where *index* is the name or index number of the custom view, to return a **CustomView** object. The following example shows the custom view named "Current Inventory."

```
ThisWorkbook.CustomViews("Current Inventory").Show
```

CustomViews Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCustomViewsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCustomViewsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCustomViewsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCustomViewsE"}

CustomViews (CustomView)

A collection of custom workbook views. Each view is represented by a **CustomView** object.

Using the CustomViews Collection

Use the **CustomViews** property to return the **CustomViews** collection. Use the **Add** method to create a new custom view and add it to the **CustomViews** collection. The following example creates a new custom view named "Summary."

```
ActiveWorkbook.CustomViews.Add "Summary", True, True
```

DataTable Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjDataTableC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjDataTableP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjDataTableM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjDataTableE"}

Represents a chart data table.

Using the DataTable Object

Use the **DataTable** property to return a **DataTable** object. The following example adds a data table with an outline border to embedded chart one.

```
With Worksheets(1).ChartObjects(1).Chart
    .HasDataTable = True
    .DataTable.HasBorderOutline = True
End With
```

HPageBreak Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjHPageBreakC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjHPageBreakP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjHPageBreakM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjHPageBreakE"}

HPageBreaks (HPageBreak)

Represents a horizontal page break. The **HPageBreak** object is a member of the **HPageBreaks** collection.

Using the HPageBreak Object

Use **HPageBreaks**(*index*), where *index* is the index number of the page break, to return an **HPageBreak** object. The following example changes the location of horizontal page break one.

```
Worksheets(1).HPageBreaks(1).Location = Worksheets(1).Range("e5")
```

HPageBreaks Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjHPageBreaksC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjHPageBreaksP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjHPageBreaksM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjHPageBreaksE"}

HPageBreaks (HPageBreak)

The collection of horizontal page breaks. Each horizontal page break is represented by an **HPageBreak** object.

Using the HPageBreaks Collection

Use the **HPageBreaks** property to return the **HPageBreaks** collection. Use the **Add** method to add a horizontal page break. The following example adds a horizontal page break above the active cell.

```
ActiveSheet.HPageBreaks.Add Before:=ActiveCell
```

VPageBreak Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjVPageBreakC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjVPageBreakP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjVPageBreakM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjVPageBreakE"}

VPageBreaks (VPageBreak)

Represents a vertical page break. The **VPageBreak** object is a member of the **VPageBreaks** collection.

Using the VPageBreak Object

Use **VPageBreaks**(*index*), where *index* is the page break index number of the page break, to return a **VPageBreak** object. The following example changes the location of vertical page break one.

```
Worksheets(1).VPageBreaks(1).Location = Worksheets(1).Range("e5")
```

VPageBreaks Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjVPageBreaksC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjVPageBreaksP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjVPageBreaksM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjVPageBreaksE"}

VPageBreaks (VPageBreak)

A collection of vertical page breaks. Each vertical page break is represented by a **VPageBreak** object.

Using the VPageBreaks Collection

Use the **VPageBreaks** property to return the **VPageBreaks** collection. Use the **Add** method to add a vertical page break. The following example adds a vertical page break to the left of the active cell.

```
ActiveSheet.VPageBreaks.Add Before:=ActiveCell
```

Hyperlink Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjHyperlinkC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjHyperlinkP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjHyperlinkM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjHyperlinkE"}

Hyperlinks (Hyperlink)

Represents a hyperlink. The **Hyperlink** object is a member of the **Hyperlinks** collection.

Using the Hyperlink Object

Use the **Hyperlink** property to return the hyperlink for a shape (a shape can have only one hyperlink). The following example activates the hyperlink for shape one.

```
Worksheets(1).Shapes(1).Hyperlink.Follow NewWindow:=True
```

A range or worksheet can have more than one hyperlink. Use **Hyperlinks**(*index*), where *index* is the hyperlink number, to return a single **Hyperlink** object. The folllowing example activates hyperlink two in the range A1:B2.

```
Worksheets(1).Range("A1:B2").Hyperlinks(2).Follow
```

Range, Shape, Worksheet

Hyperlinks Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjHyperlinksC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjHyperlinksP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjHyperlinksM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjHyperlinksE"}

Hyperlinks (Hyperlink)

Represents the collection of hyperlinks for a worksheet or range. Each hyperlink is represented by a **Hyperlink** object.

Using the Hyperlinks Collection

Use the **Hyperlinks** property to return the **Hyperlinks** collection. The following example checks the hyperlinks on worksheet one for a link that contains the word "Microsoft."

```
For Each h in Worksheets(1).Hyperlinks
   If Instr(h.Name, "Microsoft") <> 0 Then h.Follow
Next
```

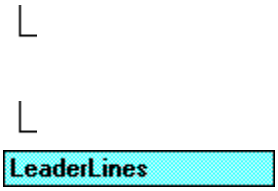
Use the **Add** method to create a hyperlink and add it to the **Hyperlinks** collection. The following example creates a new hyperlink for cell E5.

```
With Worksheets(1)
   .Hyperlinks.Add .Range("E5"), "http://www.gohere.com"
End With
```

Range, Worksheet

LeaderLines Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjLeaderLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjLeaderLinesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjLeaderLinesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjLeaderLinesE"}



Represents leader lines on a chart. Leader lines connect data labels to data points. This object isn't a collection; there's no object that represents a single leader line.

Using the LeaderLines Object

Use the **LeaderLines** property to return the **LeaderLines** object. The following example adds data labels and blue leader lines to series one on chart one.

```
With Worksheets(1).ChartObjects(1).Chart.SeriesCollection(1)
    .HasDataLabels = True
    .DataLabels.Position = xlLabelPositionBestFit
    .HasLeaderLines = True
    .LeaderLines.Border.ColorIndex = 5
End With
```

RecentFile Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjRecentFileC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjRecentFileP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjRecentFileM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjRecentFileE"}

RecentFiles (RecentFile)

Represents a file in the list of recently used files. The **RecentFile** object is a member of the **RecentFiles** collection.

Using the RecentFile Object

Use **RecentFiles**(*index*), where *index* is the file number, to return a **RecentFile** object. The following example opens file two in the list of recently used files.

```
Application.RecentFiles(2).Open
```

RecentFiles Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjRecentFilesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjRecentFilesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjRecentFilesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjRecentFilesE"}

RecentFiles (RecentFile)

Represents the list of recently used files. Each file is represented by a **RecentFile** object.

Using the RecentFiles Collection

Use the **RecentFiles** property to return the **RecentFiles** collection. The following example sets the maximum number of files in the list of recently used files.

```
Application.RecentFiles.Maximum = 6
```

PivotCache Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotCacheC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotCacheP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotCacheM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotCacheE"}

PivotCaches (PivotCache)

Represents the memory cache for a PivotTable. The **PivotCache** object is a member of the **PivotCaches** collection.

Using the PivotCache Object

Use the **PivotCache** method to return a **PivotCache** object for a PivotTable (each PivotTable has only one cache). The following example causes PivotTable one to refresh itself whenever its file is opened.

```
Worksheets(1).PivotTables(1).PivotCache.RefreshOnFileOpen = True
```

Use **PivotCaches**(*index*), where *index* is the cache number, to return a single **PivotCache** object from the **PivotCaches** collection for a workbook. The following example refreshes cache one.

```
ActiveWorkbook.PivotCaches(1).Refresh
```

PivotTable, Workbook

PivotCaches Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotCachesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotCachesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotCachesM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotCachesE"}

PivotCaches (PivotCache)

Represents the collection of PivotTable memory caches in a workbook. Each memory cache is represented by a **PivotCache** object.

Using the PivotCaches Collection

Use the **PivotCaches** method to return the **PivotCaches** collection. The following example sets the **RefreshOnFileOpen** property for all pivot caches in the active workbook.

```
For Each pc In ActiveWorkbook.PivotCaches
   pc.RefreshOnFileOpen = True
Next
```

PivotFormula Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotFormulaC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotFormulaP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotFormulaM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotFormulaE"}

Pivot Formulas (PivotFormula)

Represents a formula used to calculate results in a PivotTable.

Using the PivotFormula Object

Use **PivotFormulas**(*index*), where *index* is the formula number or string on the left side of the pivot formula, to return the **PivotFormula** object. The following example changes the index number for formula one so that it will be solved after formula two.

```
Worksheets(1).PivotTables(1).PivotFormulas(1).Index = 2
```

PivotFormulas Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPivotFormulasC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPivotFormulasP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPivotFormulasM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPivotFormulasE"}

Pivot Formulas (PivotFormula)

Represents the collection of formulas for a PivotTable. Each formula is represented by a **PivotFormula** object.

Using the PivotFormulas Collection

Use the **PivotFormulas** method to return the **PivotFormulas** collection. The following example creates a list of pivot formulas for PivotTable one.

```
For Each pf in ActiveSheet.PivotTables(1).PivotFormulas
   Cells(r, 1).Value = pf.Formula
   r = r + 1
Next
```

LinkFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjLinkFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjLinkFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjLinkFormatM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjLinkFormatE"}

Contains linked OLE object properties.

Using the LinkFormat Object

Use the **LinkFormat** property to return the **LinkFormat** object. The following example updates an OLE object in the **Shapes** collection.

```
Worksheets(1).Shapes(1).LinkFormat.Update
```

If the **Shape** object doesn't represent a linked object, the **LinkFormat** property fails.

OLEFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjOLEFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjOLEFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjOLEFormatM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjOLEFormatE"}

Contains OLE object properties.

Using the OLEFormat Object

Use the **OLEFormat** property to return the **OLEFormat** object. The following example activates an OLE object in the **Shapes** collection.

```
Worksheets(1).Shapes(1).OLEFormat.Activate
```

If the **Shape** object doesn't represent a linked or embedded object, the **OLEFormat** property fails.

Validation Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjValidationC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjValidationP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjValidationM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjValidationE"}

Validation

Represents data validation for a worksheet range.

Using the Validation Object

Use the **Validation** property to return the **Validation** object. The following example changes the data validation for cell E5.

```
Range("e5").Validation _
    .Modify xlValidateList, xlValidAlertStop, "=$A$1:$A$10"
```

Use the **Add** method to add data validation to a range and create a new **Validation** object. The following example adds data validation to cell E5.

```
With Range("e5").Validation
    .Add Type:=xlValidateWholeNumber, _
       AlertStyle:=xlValidAlertInformation, Minimum:="5", Maximum:="10"
    .InputTitle = "Integers"
    .ErrorTitle = "Integers"
    .InputMessage = "Enter an integer from five to ten"
    .ErrorMessage = "You must enter a number from five to ten"
End With
```

ControlFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjControlFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjControlFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjControlFormatM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjControlFormatE"}

Shape

ControlFormat

Contains Microsoft Excel control properties.

Using the ControlFormat Object

Use the **ControlFormat** property to return a **ControlFormat** object. The following example sets the fill range for a list box control on worksheet one.

```
Worksheets(1).Shapes(1).ControlFormat.ListFillRange = "A1:A10"
```

If the shape isn't a control, the **ControlFormat** property fails; and if the control isn't a list box, the **ListFillRange** property fails.

FormatCondition Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjFormatConditionC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjFormatConditionP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjFormatConditionM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjFormatConditionE"}

FormatConditions (FormatCondition)

Represents a conditional format. The **FormatCondition** object is a member of the **FormatConditions** collection. The **FormatConditions** collection can contain up to three conditional formats for a given range.

Using the FormatCondition Object

Use **FormatConditions**(*index*), where *index* is the index number of the conditional format, to return a **FormatCondition** object. The following example sets format properties for an existing conditional format for cells E1:E10.

```
With Worksheets(1).Range("e1:e10").FormatConditions(1)
    With .Borders
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = 6
    End With
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

Remarks

Use the **Add** method to create a new conditional format. If you try to create more than three conditional formats for a single range, the **Add** method fails. If a range has three formats, you can use the **Modify** method to change one of the formats, or you can use the **Delete** method to delete a

format and then use the **Add** method to create a new format.

Use the **Font**, **Border**, and **Interior** properties of the **FormatCondition** object to control the appearance of formatted cells. Some properties of these objects aren't supported by the conditional format object model. The properties that can be used with conditional formatting are listed in the following table.

Object	Properties
Font	**Bold**
	Color
	ColorIndex
	FontStyle
	Italic
	Strikethrough
	Underline
	The accounting underline styles cannot be used.
Border	**Bottom**
	Color
	Left
	Right
	Style
	The following border styles can be used (all others aren't supported): **xlNone**, **xlSolid**, **xlDash**, **xlDot**, **xlDashDot**, **xlDashDotDot**, **xlGray50**, **xlGray75**, and **xlGray25.**
	Top
	Weight
	The following border weights can be used (all others aren't supported): **xlWeightHairline** and **xlWeightThin.**
Interior	**Color**
	ColorIndex
	Pattern
	PatternColorIndex

FormatConditions Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjFormatConditionsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjFormatConditionsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjFormatConditionsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjFormatConditionsE"}

FormatConditions (FormatCondition)

Represents the collection of conditional formats for a single range. The **FormatConditions** collection can contain up to three conditional formats. Each format is represented by a **FormatCondition** object.

Using the FormatConditions Collection

Use the **FormatConditions** property to return a **FormatConditions** object. Use the **Add** method to create a new conditional format, and use the **Modify** method to change an existing conditional format.

The following example adds a conditional format to cells E1:E10.

```
With Worksheets(1).Range("e1:e10").FormatConditions _
        .Add(xlCellValue, xlGreater, "=$a$1")
    With .Borders
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = 6
    End With
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

Remarks

If you try to create more than three conditional formats for a single range, the **Add** method fails. If a range has three formats, you can use the **Modify** method to change one of the formats, or you can

use the **Delete** method to delete a format and then use the **Add** method to create a new format. For more information about conditional formats, see the **FormatCondition** object.

AutoCorrect Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAutoCorrectC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAutoCorrectP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAutoCorrectM"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAutoCorrectE"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjAutoCorrectE"}

AutoCorrect

Contains Microsoft Excel AutoCorrect attributes (capitalization of names of days, correction of two initial capital letters, automatic correction list, and so on).

Using the AutoCorrect Object

Use the **AutoCorrect** property to return the **AutoCorrect** object. The following example sets Microsoft Excel to correct words that begin with two initial capital letters.

```
With Application.AutoCorrect
    .TwoInitialCapitals = True
    .ReplaceText = True
End With
```

Corners Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjCornersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjCornersX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCornersP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCornersM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCornersE"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjCornersE"}

Corners

Represents the corners of a 3-D chart. This object isn't a collection.

Using the Corners Object

Use the **Corners** property to return the **Corners** object. The following example selects the corners of chart one.

```
Charts(1).Corners.Select
```

If the chart isn't a 3-D chart, the **Corners** property fails.

OLEObject Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjOLEObjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjOLEObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjOLEObjectP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjOLEObjectM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjOLEObjectE"}

OLEObjects (OLEObject)

Represents an ActiveX control or a linked or embedded OLE object on a worksheet. The **OLEObject** object is a member of the **OLEObjects** collection. The **OLEObjects** collection contains all the OLE objects on a single worksheet.

Using the OLEObject Object

Use **OLEObjects**(*index*), where *index* is the name or number of the object, to return an **OLEObject** object. The following example deletes OLE object one on Sheet1.

```
Worksheets("sheet1").OLEObjects(1).Delete
```

The following example deletes the OLE object named "ListBox1."

```
Worksheets("sheet1").OLEObjects("ListBox1").Delete
```

Remarks

The properties and methods of the **OLEObject** object are duplicated on each ActiveX control on a worksheet. This enables Visual Basic code to gain access to these properties by using the control's name. The following example selects the check box control named "MyCheckBox," aligns it with the active cell, and then activates the control.

```
With MyCheckBox
    .Value = True
    .Top = ActiveCell.Top
    .Activate
End With
```

For more information, see Using ActiveX controls on sheets.

Chart, **Worksheet**

OLEObjects Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjOLEObjectsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"xlobjOLEObjectsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjOLEObjectsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjOLEObjectsM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjOLEObjectsE"}

OLEObjects (OLEObject)

A collection of all the **OLEObject** objects on the specified worksheet. Each **OLEObject** object represents an ActiveX control or a linked or embedded OLE object.

Using the OLEObjects Collection

Use the **OLEObjects** method to return the **OLEObjects** collection. The following example hides all the OLE objects on worksheet one.

```
Worksheets(1).OLEObjects.Visible = False
```

Use the **Add** method to create a new OLE object and add it to the **OLEObjects** collection. The following example creates a new OLE object representing the bitmap file Arcade.bmp and adds it to worksheet one.

```
Worksheets(1).OLEObjects.Add FileName:="arcade.bmp"
```

The following example creates a new ActiveX control (a list box) and adds it to worksheet one.

```
Worksheets(1).OLEObjects.Add ClassType:="Forms.ListBox.1"
```

For more information, see Using ActiveX controls on sheets.

Adjustments Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjAdjustmentsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjAdjustmentsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjAdjustmentsM"}

Shapes (Shape)

└

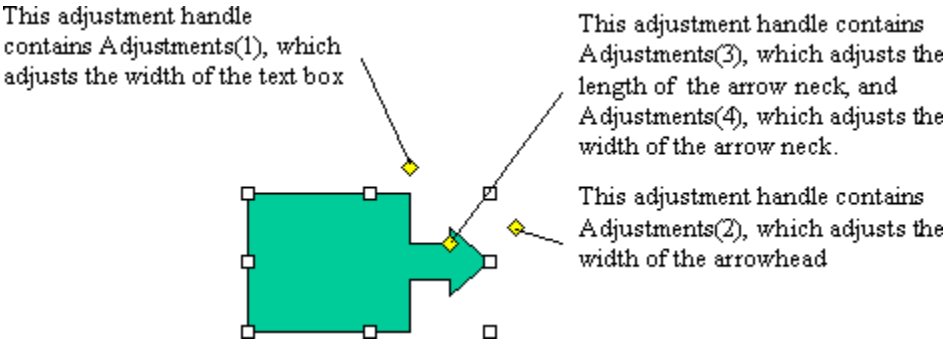
Adjustments

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 = Worksheets(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":"xlobjCalloutFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjCalloutFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjCalloutFormatM"}

Shapes (Shape)

CalloutFormat

Contains properties and methods that apply to line callouts.

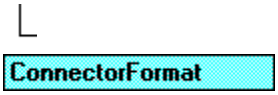
Using the CalloutFormat Object

Use the **Callout** property to return a **CalloutFormat** object. The following example specifies the following attributes of shape three (a line callout) on `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 = Worksheets(1)
With myDocument.Shapes(3).Callout
    .Accent = True
    .Angle = msoCalloutAngle30
    .Border = False
    .PresetDrop msoCalloutDropTop
    .Type = msoCalloutThree
End With
```

ConnectorFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjConnectorFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjConnectorFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjConnectorFormatM"}



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 = Worksheets(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
Setc c = s.AddConnector(msoConnectorCurve, 0, 0, 0, 0)
With c.ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=1
    c.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 ActiveSheet
    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.Characters.Text = j
        End With
    Next j
End With
```

GroupShapes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjGroupShapesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjGroupShapesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjGroupShapesM"}

Shapes (Shape)

L

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.

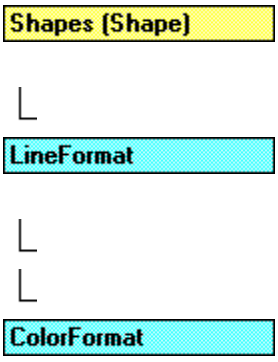
Using The GroupShapes Collection

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 = Worksheets(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":"xlobjLineFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjLineFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjLineFormatM"}



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 = Worksheets(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
```

PictureFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjShapeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjShapeM"}

Shapes (Shape)

└

PictureFormat

Contains properties and methods that apply to pictures and OLE objects. The **LinkFormat** object contains properties and methods that apply to linked OLE objects only. The **OLEFormat** object contains properties and methods that apply to OLE objects whether or not they're linked.

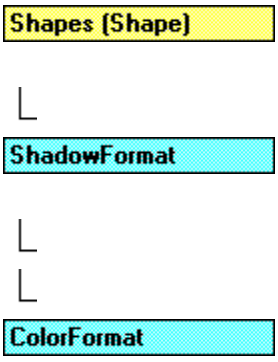
Using the PictureFormat Object

Use the **PictureFormat** property to return a **PictureFormat** object. The following example sets the brightness, contrast, and color transformation for shape one on `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 = Worksheets(1)
With myDocument.Shapes(1).PictureFormat
    .Brightness = 0.3
    .Contrast = 0.7
    .ColorType = msoPictureGrayScale
    .CropBottom = 18
```

ShadowFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjShadowFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjShadowFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjShadowFormatM"}



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 = Worksheets(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":"xlobjShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjShapeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjShapeM"}

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); and the **Shape** object, which represents a single shape on a document. If you want to work with several shapes 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.
- Return a shape within the selection.
- Return the shapes attached to the ends of a connector.
- Return a newly created freeform.
- Return a single shape from within a group.
- Return a newly formed group of shapes.

Returning an Existing Shape

Use **Shapes**(*index*), where *index* is the shape name or the index number, to return a **Shape** object that represents a shape. The following example horizontally flips shape one and the shape named "Rectangle 1" on `myDocument`.

```
Set myDocument = Worksheets(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 = Worksheets(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 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 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 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.

ShapeRange Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjShapeRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjShapeRangeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjShapeRangeM"}

ShapeRange (Shape)

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 = Worksheets(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 = Worksheets(1)
Set myRange = myDocument.Shapes.Range(Array("Oval 4", "Rectangle 5"))
myRange.Fill.Patterned msoPatternHorizontalBrick
```

Although you can use the **Range** property 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)
```

ShapeNode Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjShapeNodeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjShapeNodeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjShapeNodeM"}

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 = Worksheets(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":"xlobjShapeNodesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjShapeNodesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjShapeNodesM"}

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 = Worksheets(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 = Worksheets(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 = Worksheets(1)
With myDocument.Shapes(3)
    If .Nodes(1).EditingType = msoEditingCorner Then
        .Nodes.SetEditingType 1, msoEditingSmooth
    End If
End With
```

TextEffectFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjTextEffectFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjTextEffectFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjTextEffectFormatM"}

TextEffectFormat

Contains properties and methods that apply to WordArt objects.

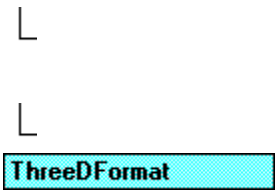
Using the TextEffectFormat Object

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 = Worksheets(1)
With myDocument.Shapes(1).TextEffect
    .FontName = "Courier New"
    .FontBold = True
    .FontItalic = True
End With
```

ThreeDFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjThreeDFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjThreeDFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjThreeDFormatM"}



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 = Worksheets(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.

FreeformBuilder Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjFreeformBuilderC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjFreeformBuilderP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjFreeformBuilderM"}

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 = Worksheets(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
```

ColorFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjColorFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjColorFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjColorFormatM"}

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
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**
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**
Color of the sides of an extruded object	**ExtrusionColor**	**ThreeDFormat**

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 = Worksheets(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
```

Shapes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjShapesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjShapesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjShapesM"}

A collection of all the **Shape** objects on the specified sheet. 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 = Worksheets(1)
myDocument.Shapes.SelectAll
```

Note If you want to do something (like delete or set a property) to all the shapes on a sheet at the same time, select all the shapes and then use the **ShapeRange** property on the selection to create a **ShapeRange** object that contains all the shapes on the sheet, and then apply the appropriate property or method to the **ShapeRange** object.

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 = Worksheets(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 = Worksheets(1)
myDocument.Shapes.Range(Array(1, 3)).Fill.Patterned
msoPatternHorizontalBrick
```

TextFrame Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjTextFrameC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjTextFrameP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjTextFrameM"}

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.

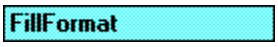
```
Set myDocument = Worksheets(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 250,
140).TextFrame
    .Characters.Text = "Here is some test text"
    .MarginBottom = 10
    .MarginLeft = 10
    .MarginRight = 10
    .MarginTop = 10
End With
```

text frame

The area within a shape that can contain text.

FillFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjFillFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjFillFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjFillFormatM"}



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 = Worksheets(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.

Adjustments, **CalloutFormat**, **ConnectorFormat**, **FillFormat**, **GroupShapes**, **LineFormat**, **LinkFormat**, **OLEFormat**, **PictureFormat**, **ShadowFormat**, **ShapeNodes**, **ShapeRange**, **TextEffectFormat**, **TextFrame**, **ThreeDFormat**

FillFormat, **LineFormat**, **ShadowFormat**, **ThreeDFormat**

Adjustments, **CalloutFormat**, **ConnectorFormat**, **FillFormat**, **GroupShapes**, **LineFormat**, **LinkFormat**, **OLEFormat**, **PictureFormat**, **ShadowFormat**, **Shape**, **ShapeNodes**, **ShapeRange**, **TextEffectFormat**, **TextFrame**, **ThreeDFormat**

WorksheetFunction Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjWorksheetFunctionC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjWorksheetFunctionP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjWorksheetFunctionM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjWorksheetFunctionE"}

WorksheetFunction

Used as a container for Microsoft Excel worksheet functions that can be called from Visual Basic.

Using the WorksheetFunction Object

Use the **WorksheetFunction** property to return the **WorksheetFunction** object. The following example displays the result of applying the **Min** worksheet function to the range A1:A10.

```
Set myRange = Worksheets("Sheet1").Range("A1:C10")
answer = Application.WorksheetFunction.Min(myRange)
MsgBox answer
```

Remarks

In previous versions of Microsoft Excel, worksheet functions were contained by the **Application** object.

ChartColorFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartColorFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartColorFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartColorFormatM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartColorFormatE"}

ChartFillFormat

└

ChartColorFormat

Used only with charts. Represents the color of a one-color object or the foreground or background color of an object with a gradient or patterned fill.

Using the ChartColorFormat Object

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

To return a ChartColorFormat object that represents this	Use this property	With this object
Background fill color (used in a shaded or patterned fill)	**BackColor**	**ChartFillFormat**
Foreground fill color (or just the fill color for a solid fill)	**ForeColor**	**ChartFillFormat**

ChartFillFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjChartFillFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjChartFillFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjChartFillFormatM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjChartFillFormatE"}

ChartFillFormat

ChartColorFormat

Used only with charts. Represents fill formatting for chart elements.

Using the ChartFillFormat Object

Use the **Fill** property to return a **ChartFillFormat** object. The following example sets the foreground color, background color, and gradient for the chart area fill on chart one.

```
With Charts(1).ChartArea.Fill
    .Visible = True
    .ForeColor.SchemeColor = 15
    .BackColor.SchemeColor = 17
    .TwoColorGradient Style:=msoGradientHorizontal, Variant:=1
End With
```

AxisTitle, **ChartArea**, **ChartTitle**, **DataLabel**, **DataLabels**, **DownBars**, **Floor**, **Legend**, **LegendKey**, **PlotArea**, **Point**, **Series**, **Shape**, **ShapeRange**, **UpBars**, **Walls**

Phonetic Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"xlobjPhoneticC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"xlobjPhoneticP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"xlobjPhoneticM"} {ewc HLP95EN.DLL, DYNALINK, "Events":"xlobjPhoneticE"}

This object is not used in U.S. English Microsoft Excel.

DocumentProperty Object (Microsoft Excel)

DocumentProperties (Document Property)

Each workbook has a collection of built-in document properties and a collection of custom document properties. Each collection is represented by a **DocumentProperties** object, and each collection contains individual **DocumentProperty** objects.

Use the **BuiltinDocumentProperties** property to return the collection of built-in document properties, and use the **CustomDocumentProperties** property to return the collection of custom document properties. Use the **Item** property to return a single member of the collection.

The following example sets the value of the built-in document property named "Title."

```
ActiveWorkbook.BuiltinDocumentProperties.Item("Title") _
    .Value = "Year-End Sales Results"
```

DocumentProperties Collection Object (Microsoft Excel)

DocumentProperties (Document Property)

Each workbook has a collection of built-in document properties and a collection of custom document properties. Each collection is represented by a **DocumentProperties** object, and each collection contains individual **DocumentProperty** objects.

Use the **BuiltinDocumentProperties** property to return the collection of built-in document properties, and use the **CustomDocumentProperties** property to return the collection of custom document properties.

The following example displays the names of the built-in document properties as a list on worksheet one.

```
rw = 1
Worksheets(1).Activate
For Each p In ActiveWorkbook.BuiltinDocumentProperties
    Cells(rw, 1).Value = p.Name
    rw = rw + 1
Next
```