

# Microsoft Office Web Components Object Models

Select an object model from the following list.

[ChartSpace Object Model](#)

[Data Source Control Object Model](#)

[PivotTable List Object Model](#)

[Spreadsheet Object Model](#)

--

## **What's New for Microsoft Office Web Components Developers**

Extensive changes have been made to the Microsoft Office Web Components Visual Basic object model to support new and improved features in the application. The following sections provide information about major features that are new in the Office Web Components object model.

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

Click one of the following entries to jump to a section.

[New Objects](#)

[New Events \(Alphabetical\)](#)

[New Events \(By Object\)](#)

[New Methods \(Alphabetical\)](#)

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

[New Properties \(Alphabetical\)](#)

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

## New Objects

The following table lists the new objects in the Microsoft Office Web Components Visual Basic object model.

Objects
<a href="#">ByRef</a>
<a href="#">ChAxes</a>
<a href="#">ChAxis</a>
<a href="#">ChBorder</a>
<a href="#">ChCategoryLabel</a>
<a href="#">ChCategoryLabels</a>
<a href="#">ChChart</a>
<a href="#">ChChartDraw</a>
<a href="#">ChChartField</a>
<a href="#">ChChartFields</a>
<a href="#">ChCharts</a>
<a href="#">ChDataLabel</a>
<a href="#">ChDataLabels</a>
<a href="#">ChDataLabelsCollection</a>
<a href="#">ChDropZone</a>
<a href="#">ChErrorBars</a>
<a href="#">ChErrorBarsCollection</a>
<a href="#">ChFont</a>
<a href="#">ChFormatMap</a>
<a href="#">ChGridlines</a>
<a href="#">ChInterior</a>
<a href="#">ChLegend</a>
<a href="#">ChLegendEntries</a>
<a href="#">ChLegendEntry</a>
<a href="#">ChLine</a>
<a href="#">ChMarker</a>
<a href="#">ChPlotArea</a>
<a href="#">ChPoint</a>
<a href="#">ChPoints</a>
<a href="#">ChScaling</a>
<a href="#">ChSegment</a>
<a href="#">ChSegmentBoundary</a>

ChSegments
ChSeries
ChSeriesCollection
ChSurface
ChTitle
ChTrendline
ChTrendlines
ChUserDefinedSelection
Coordinate
Font
Heading
Headings
Name
Names
OCCommand
OCCommands
OWCLanguageSettings
PivotAxisMember
PivotAxisMembers
PivotColumnMember
PivotColumnMembers
PivotHyperlink
PivotMemberProperties
PivotMemberProperty
PivotPageMember
PivotPageMembers
PivotResultAxis
PivotResultColumnAxis
PivotResultDataAxis
PivotResultField
PivotResultFilterAxis
PivotResultGroupAxis
PivotResultGroupField
PivotResultGroupFields
PivotResultLabel
PivotResultMemberProperties
PivotResultMemberProperty

**PivotResultPageAxis**

---

**PivotResultRowAxis**

---

**PivotRowMember**

---

**PivotRowMembers**

---

**Sheets**

---

**Window**

---

**Windows**

---

**Workbook**

---

**Workbooks**

---

**Worksheets**

---

## New Events (Alphabetical)

The following table lists the new events (alphabetically) in the Microsoft Office Web Components Visual Basic type library.

Events
<a href="#">AfterDelete</a>
<a href="#">AfterFinalRender</a>
<a href="#">AfterInsert</a>
<a href="#">AfterLayout</a>
<a href="#">AfterRender</a>
<a href="#">AfterUpdate</a>
<a href="#">BeforeContextMenu</a>
<a href="#">BeforeDelete</a>
<a href="#">BeforeInitialBind</a>
<a href="#">BeforeInsert</a>
<a href="#">BeforeKeyDown</a>
<a href="#">BeforeKeyPress</a>
<a href="#">BeforeKeyUp</a>
<a href="#">BeforeOverwrite</a>
<a href="#">BeforeQuery</a>
<a href="#">BeforeRender</a>
<a href="#">BeforeScreenTip</a>
<a href="#">BeforeUpdate</a>
<a href="#">CommandBeforeExecute</a>
<a href="#">CommandChecked</a>
<a href="#">CommandEnabled</a>
<a href="#">CommandExecute</a>
<a href="#">CommandTipText</a>
<a href="#">DataChange</a>
<a href="#">Dirty</a>
<a href="#">Focus</a>
<a href="#">Initialize</a>
<a href="#">LoadCompleted</a>
<a href="#">MouseWheel</a>
<a href="#">OnConnect</a>
<a href="#">OnDisconnect</a>
<a href="#">RecordExit</a>

**RecordsetSaveProgress**

---

**SheetActivate**

---

**SheetCalculate**

---

**SheetChange**

---

**SheetDeactivate**

---

**SheetFollowHyperlink**

---

**Undo**

---

## New Events (By Object)

The following table lists the new events (by object) in the Microsoft Office Web Components Visual Basic type library.

Objects	Events
ChartSpace	AfterFinalRender
	AfterLayout
	AfterRender
	BeforeContextMenu
	BeforeKeyDown
	BeforeKeyPress
	BeforeKeyUp
	BeforeRender
	BeforeScreenTip
	CommandBeforeExecute
	CommandChecked
	CommandEnabled
	CommandExecute
	CommandTipText
	MouseWheel
DataSourceControl	AfterDelete
	AfterInsert
	AfterUpdate
	BeforeDelete
	BeforeInitialBind
	BeforeInsert
	BeforeOverwrite
	BeforeUpdate
	Dirty
	Focus
	RecordExit
	RecordsetSaveProgress
PivotTable	Undo
	BeforeContextMenu



	<b>BeforeKeyDown</b>
	<b>BeforeKeyPress</b>
	<b>BeforeKeyUp</b>
	<b>BeforeQuery</b>
	<b>BeforeScreenTip</b>
	<b>CommandBeforeExecute</b>
	<b>CommandChecked</b>
	<b>CommandEnabled</b>
	<b>CommandExecute</b>
	<b>CommandTipText</b>
	<b>DataChange</b>
	<b>MouseWheel</b>
	<b>OnConnect</b>
	<b>OnDisconnect</b>
<b>Spreadsheet</b>	<b>BeforeContextMenu</b>
	<b>BeforeKeyDown</b>
	<b>BeforeKeyPress</b>
	<b>BeforeKeyUp</b>
	<b>CommandBeforeExecute</b>
	<b>CommandChecked</b>
	<b>CommandEnabled</b>
	<b>CommandExecute</b>
	<b>CommandTipText</b>
	<b>Initialize</b>
	<b>LoadCompleted</b>
	<b>MouseWheel</b>
	<b>SheetActivate</b>
	<b>SheetCalculate</b>
	<b>SheetChange</b>
	<b>SheetDeactivate</b>
	<b>SheetFollowHyperlink</b>

## New Methods (Alphabetical)

The following table lists the new methods (alphabetically) in the Microsoft Office Web Components Visual Basic type library.

Methods
<a href="#">Activate</a>
<a href="#">AddCalculatedField</a>
<a href="#">AddCalculatedTotal</a>
<a href="#">AddCustomGroupField</a>
<a href="#">AddCustomGroupMember</a>
<a href="#">AddFieldSet</a>
<a href="#">ApplyFilter</a>
<a href="#">AutoFit</a>
<a href="#">BeginObject</a>
<a href="#">BorderAround</a>
<a href="#">CalculateFull</a>
<a href="#">CopyFromRecordset</a>
<a href="#">DeleteCustomGroupMember</a>
<a href="#">DeleteField</a>
<a href="#">DeleteFieldSet</a>
<a href="#">DrawEllipse</a>
<a href="#">DrawLine</a>
<a href="#">DrawPolygon</a>
<a href="#">DrawPolyLine</a>
<a href="#">DrawRectangle</a>
<a href="#">DrawText</a>
<a href="#">DropZones</a>
<a href="#">DuplicateFormat</a>
<a href="#">EndObject</a>
<a href="#">EuroConvert</a>
<a href="#">Evaluate</a>
<a href="#">ExportXML</a>
<a href="#">FillDown</a>
<a href="#">FillRight</a>
<a href="#">FindNext</a>
<a href="#">FindPrevious</a>
<a href="#">GetPicture</a>

Group
HideDetails
Insert
Item
LargeScroll
Move
MoveDetailLeft
MoveDetailTop
MoveLeft
MoveTop
Nz
PointsToScreenPixelsX
PointsToScreenPixelsY
Protect
RectIntersect
RectUnion
Repaint
Reset
ResetColors
ScrollIntoView
SetOneColorGradient
SetPatterned
SetPresetGradient
SetRootRecordset
SetSolid
SetSpreadsheetData
SetTextured
SetTwoColorGradient
ShowAbout
ShowContextMenu
ShowDetails
ShowHelp
SmallScroll
SortAscending
SortDescending
TextHeight
TextWidth

**ToggleFilter**

---

**Ungroup**

---

**Unprotect**

---

**ValueToPoint**

---

## New Methods (By Object)

The following table lists the new methods (by object) in the Microsoft Office Web Components Visual Basic type library.

Objects	Methods
ChartSpace	DropZones
	GetPicture
	Repaint
	SetSpreadsheetData
	ShowContextMenu
	ShowHelp
ChAxis	ValueToPoint
ChChart	DuplicateFormat
	SetSpreadsheetData
ChChartDraw	BeginObject
	DrawEllipse
	DrawLine
	DrawPolygon
	DrawPolyLine
	DrawRectangle
	DrawText
	EndObject
	TextHeight
	TextWidth
ChInterior	SetOneColorGradient
	SetPatterned
	SetPresetGradient
	SetSolid
	SetTextured
	SetTwoColorGradient
ChSeries	Group
	Ungroup
	ValueToPoint

DataPage	ApplyFilter
	SortAscending
	SortDescending
	ToggleFilter
DataSourceControl	EuroConvert
	ExportXML
	Nz
	Reset
	SetRootRecordset
Heading	Reset
Names	Item
PivotAxisMember	HideDetails
	ShowDetails
PivotCell	MoveDetailTop
PivotColumnMember	HideDetails
	MoveDetailLeft
	ShowDetails
PivotData	HideDetails
	MoveLeft
	MoveTop
	ShowDetails
PivotField	AddCustomGroupMember
	DeleteCustomGroupMember
PivotFieldSet	AddCalculatedField
	AddCustomGroupField
	DeleteField
PivotPageMember	HideDetails
	ShowDetails
PivotRowMember	HideDetails
	ShowDetails
PivotTable	ShowAbout
	ShowContextMenu

	ShowHelp
PivotView	AddCalculatedTotal AddFieldSet DeleteFieldSet
Range	Activate AutoFit BorderAround CopyFromRecordset FillDown FillRight FindNext FindPrevious Insert
Sheets	Move
Spreadsheet	CalculateFull Evaluate RectIntersect RectUnion Repaint ShowAbout ShowContextMenu ShowHelp
Window	LargeScroll PointsToScreenPixelsX PointsToScreenPixelsY ScrollIntoView SmallScroll
Workbook	Protect ResetColors Unprotect
Worksheet	Activate Evaluate Move

	Protect
	Unprotect
Worksheets	Move



## New Properties (Alphabetical)

The following table lists the new properties (alphabetically) in the Microsoft Office Web Components Visual Basic type library.

Properties
<a href="#">ActiveObject</a>
<a href="#">ActiveWindow</a>
<a href="#">ActiveWorkbook</a>
<a href="#">AllIncludeExclude</a>
<a href="#">AllMember</a>
<a href="#">AllowAdditions</a>
<a href="#">AllowCustomOrdering</a>
<a href="#">AllowDeletions</a>
<a href="#">AllowEdits</a>
<a href="#">AllowFormattingColumns</a>
<a href="#">AllowFormattingRows</a>
<a href="#">AllowHeadingRename</a>
<a href="#">AllowLayoutEvents</a>
<a href="#">AllowMultiFilter</a>
<a href="#">AllowPointRenderEvents</a>
<a href="#">AllowRenderEvents</a>
<a href="#">AllowScreenTipEvents</a>
<a href="#">AlternateRowColor</a>
<a href="#">AlwaysIncludeInCube</a>
<a href="#">AmbientLightIntensity</a>
<a href="#">Application</a>
<a href="#">AspectRatio</a>
<a href="#">Axis</a>
<a href="#">BackWall</a>
<a href="#">Begin</a>
<a href="#">Bookmark</a>
<a href="#">Bottom</a>
<a href="#">Build</a>
<a href="#">ButtonBorder</a>
<a href="#">ButtonFont</a>
<a href="#">ButtonInterior</a>
<a href="#">Calculation</a>

CalculationVersion

---

CategoryLabels

---

CellsEx

---

ChartDepth

---

ChartFields

---

Checked

---

ChildAxisMembers

---

ChildColumnMembers

---

ChildPageMembers

---

ChildRowMembers

---

ColorIndex

---

Colors

---

ColumnHeadings

---

Commands

---

CompareMemberCaptionsBy

---

CompareOrderedMembersBy

---

ConnectionFile

---

Control

---

CurrentArray

---

CurrentCell

---

CustomGroupMembers

---

CustomGroupType

---

DashStyle

---

Data

---

DataSourceName

---

DefaultColor

---

DefaultMember

---

DesignMode

---

DetailColumnCount

---

DetailLeftOffset

---

DetailRowCount

---

DetailsExpanded

---

DetailTopOffset

---

DirectionalLightInclination

---

DirectionalLightIntensity

---

DirectionalLightRotation

---

DisplayAlert

---

**DisplayCalculatedMembers**

---

**DisplayColumnHeadings**

---

**DisplayCustomHeadings**

---

**DisplayDesignTimeUI**

---

**DisplayFieldButtons**

---

**DisplayHeadings**

---

**DisplayIn**

---

**DisplayInFieldList**

---

**DisplayOfficeLogo**

---

**DisplayRowHeadings**

---

**DisplayScreenTips**

---

**DisplayTotal**

---

**DisplayWorkbookTabs**

---

**DisplayZeros**

---

**Divisions**

---

**DrawType**

---

**EditMode**

---

**EnableResize**

---

**End**

---

**ExcludedMembers**

---

**ExpandDetails**

---

**ExpandMembers**

---

**Expression**

---

**ExtrudeAngle**

---

**FillType**

---

**FilterCaption**

---

**FilterContext**

---

**FilterCrossJoins**

---

**FilterFunctionValue**

---

**FilterOn**

---

**FilterOnScope**

---

**FindAxisMember**

---

**FindColumnMember**

---

**FindMember**

---

**FindPageMember**

---

**FindRowMember**

---

**Floor**

---

Format
FormatMap
FormulaArray
GapDepth
GradientDegree
GradientStyle
GradientVariant
GridlineColor
GridlineColorIndex
GroupedBackColor
GroupedFont
GroupedForeColor
GroupedHAlignment
GroupedHeight
GroupEnd
GroupField
GroupFields
GroupingTotalFunction
GroupingType
GroupingUnit
GroupingUnitType
GroupStart
HasAbsoluteLabels
HasArray
HasAutoAspectRatio
HasAutoChartDepth
HasAutoDivisions
HasDiscreteDivisions
HasMultipleCharts
HasPassiveAlerts
HasPlotDetails
HasRuntimeSelection
HasSelectionMarks
HasSeriesByRows
HasUnifiedScales
HeaderBackColor
HeaderFont

HeaderForeColor  
HeaderHAlignment

HeaderHeight  
HorizontalAlignment  
Hwnd

id  
Inclination

IncludedMembers  
InternalPivotTable

International  
IsConsistent

IsDataBound  
IsDirty

IsHyperlink  
IsValid

ItemCount  
LanguageID

LanguagePreferredForEditing  
LanguageSettings

Left2  
LeftOffset

Level  
LevelCount

LightNormal  
Member

MemberCaptions  
MemberProperties

MemberPropertiesOrder  
MemberProperty

MergeCells  
Miter

Names  
Next

Offline  
OfflinePublication

OfflineSource  
OfflineType

Offset
OrderedMembers
PageAxis
PageMember
ParentAxisMember
ParentColumnMember
ParentLabel
ParentPageMember
ParentRowMember
Path
Pattern
PercentComplete
Period
Perspective
PivotAxis
PivotObject
PlotAllAggregates
PrefixCharacter
PresetGradientType
PresetTexture
Previous
PrintQuality3D
ProjectionMode
PropertyCaptionFont
PropertyCaptionHAlignment
PropertyCaptionWidth
PropertyHeight
PropertyValueFont
PropertyValueHAlignment
PropertyValueWidth
ProtectContents
ProtectionMode
ProtectStructure
ProviderFormattedValue
RangeSelection
ReadingOrder
RecordSelector

RefersTo
RefersToLocal
RefersToRange
Right
Rotation
RowHeadings
ScrollColumn
ScrollRow
Segments
SelectedSheets
Sheets
ShowAs
ShowAsValue
SideWall
SolveOrder
SourceAxis
SourceColumnAxis
SourceDataAxis
SourceField
SourceFilterAxis
SourceLabel
SourceMember
SourcePageAxis
SourceRowAxis
StandardHeight
StandardWidth
Status
SubtotalLabelBackColor
SubtotalLabelFont
SubtotalLabelForeColor
SubtotalLabelHAlignment
TabRatio
TextureFormat
TextureName
TexturePlacement
TextureStackUnit
Thickness

**TickLabelUnitType**

**TickMarkUnitType**

**TimeValue**

**TipText**

**Toolbar**

**Top2**

**TopOffset**

**TotalColumnMember**

**TotalPageMember**

**TotalRowHeight**

**TotalRowMember**

**UsableHeight**

**UsableWidth**

**UserMode**

**UseStandardHeight**

**UseStandardWidth**

**UseXMLData**

**Value2**

**ValueType**

**VerticalAlignment**

**View**

**ViewOnlyMode**

**ViewportLeft**

**ViewportTop**

**WatermarkBorder**

**WatermarkFont**

**WatermarkInterior**

**WindowNumber**

**Windows**

**Workbooks**

**Worksheet**

**Worksheets**

**XMLDataTarget**

**XMLLocation**

**XMLURL**

**ZOrder**



## New Properties (By Object)

The following table lists the new properties (by object) in the Microsoft Office Web Components Visual Basic type library.

Objects	Properties
<a href="#">AutoFilter</a>	<a href="#">Application</a>
<a href="#">Border</a>	<a href="#">Application</a> <a href="#">ColorIndex</a>
<a href="#">Borders</a>	<a href="#">Application</a> <a href="#">ColorIndex</a>
<a href="#">ChartSpace</a>	<a href="#">AllowLayoutEvents</a> <a href="#">AllowPointRenderEvents</a> <a href="#">AllowRenderEvents</a> <a href="#">AllowScreenTipEvents</a> <a href="#">Bottom</a> <a href="#">Commands</a> <a href="#">DataSourceName</a> <a href="#">DisplayFieldButtons</a> <a href="#">DisplayOfficeLogo</a> <a href="#">DisplayScreenTips</a> <a href="#">HasMultipleCharts</a> <a href="#">HasPassiveAlerts</a> <a href="#">HasPlotDetails</a> <a href="#">HasRuntimeSelection</a> <a href="#">HasSelectionMarks</a> <a href="#">HasSeriesByRows</a> <a href="#">HasUnifiedScales</a> <a href="#">InternalPivotTable</a> <a href="#">International</a> <a href="#">IsDirty</a> <a href="#">LanguageSettings</a> <a href="#">PlotAllAggregates</a> <a href="#">PrintQuality3D</a> <a href="#">Right</a>

	<b>Toolbar</b> <b>ViewOnlyMode</b>
<b>ChAxis</b>	<b>Bottom</b> <b>CategoryLabels</b> <b>GroupingTotalFunction</b> <b>GroupingType</b> <b>GroupingUnit</b> <b>GroupingUnitType</b> <b>Right</b> <b>TickLabelUnitType</b> <b>TickMarkUnitType</b>
<b>ChBorder</b>	<b>DashStyle</b>
<b>ChCategoryLabel</b>	<b>Bottom Level</b> <b>ParentLabel</b> <b>PivotObject</b> <b>Right</b>
<b>ChCategoryLabels</b>	<b>ItemCount</b> <b>LevelCount</b> <b>PivotAxis</b>
<b>ChChart</b>	<b>AmbientLightIntensity</b> <b>AspectRatio</b> <b>Bottom</b> <b>ChartDepth</b> <b>DirectionalLightInclination</b> <b>DirectionalLightIntensity</b> <b>DirectionalLightRotation</b> <b>ExtrudeAngle</b> <b>GapDepth</b> <b>HasAutoAspectRatio</b> <b>HasAutoChartDepth</b> <b>Inclination</b> <b>LightNormal</b> <b>Perspective</b>

	ProjectionMode
	Right
	Rotation
ChChartDraw	DrawType
ChChartField	Bottom
	PivotObject
	Right
ChDataLabel	Bottom
	Right
ChDataLabels	Bottom
	Right
ChDropZone	Bottom
	ButtonBorder
	ButtonFont
	ButtonInterior
	ChartFields
	Right
	WatermarkBorder
	WatermarkFont
	WatermarkInterior
ChErrorBars	Bottom
	Right
ChFormatMap	Segments
ChInterior	DefaultColor
	FillType
	GradientDegree
	GradientStyle
	GradientVariant
	Pattern
	PresetGradientType
	PresetTexture
	TextureFormat
	TextureName

	TexturePlacement
	TextureStackUnit
ChLegend	Bottom
	Right
ChLegendEntry	Bottom
	PivotObject
	Right
ChLine	DashStyle
	Miter
ChPlotArea	BackWall
	Bottom
	Floor
	Right
	SideWall
ChPoint	Bottom
	PivotObject
	Right
ChSegment	Begin
	Divisions
	End
	HasAbsoluteLabels
	HasAutoDivisions
	HasDiscreteDivisions
ChSegmentBoundary	ValueType
ChSeries	Bottom
	FormatMap
	PivotObject
	Right
	Thickness
	TipText
	ZOrder
ChSeriesCollection	PivotAxis

<b>ChSurface</b>	<b>Thickness</b>
<b>ChTitle</b>	<b>Bottom</b>
	<b>Right</b>
<b>ChTrendline</b>	<b>Bottom</b>
	<b>Period</b>
	<b>Right</b>
<b>ChUserDefinedSelection</b>	<b>id</b>
<b>Criteria</b>	<b>Application</b>
<b>DataSourceControl</b>	<b>ConnectionFile</b>
	<b>IsDirty</b>
	<b>Offline</b>
	<b>OfflinePublication</b>
	<b>OfflineSource</b>
	<b>OfflineType</b>
	<b>UseXMLData</b>
	<b>XMLDataTarget</b>
	<b>XMLLocation</b>
<b>DSCEventInfo</b>	<b>DisplayAlert</b>
	<b>PercentComplete</b>
	<b>Status</b>
<b>ElementExtension</b>	<b>Format</b>
<b>Filter</b>	<b>Application</b>
<b>Filters</b>	<b>Application</b>
<b>Font</b>	<b>Application</b>
	<b>ColorIndex</b>
<b>GroupLevel</b>	<b>AllowAdditions</b>
	<b>AllowDeletions</b>
	<b>AllowEdits</b>
	<b>AlternateRowColor</b>
	<b>RecordSelector</b>
<b>Heading</b>	<b>Application</b>
<b>Headings</b>	<b>Application</b>
<b>Hyperlink</b>	<b>Application</b>
<b>Interior</b>	<b>Application</b>

## ColorIndex

<b>Name</b>	<b>Application</b> <b>RefersTo</b> <b>RefersToLocal</b> <b>RefersToRange</b>
<b>Names</b>	<b>Application</b>
<b>OCCommand</b>	<b>Checked</b> <b>TipText</b>
<b>OWCLanguageSettings</b>	<b>Application</b> <b>LanguageID</b> <b>LanguagePreferredForEditing</b>
<b>Pane</b>	<b>Application</b>
<b>Panes</b>	<b>Application</b>
<b>ParameterValue</b>	<b>Expression</b>
<b>PivotAggregate</b>	<b>ProviderFormattedValue</b> <b>ShowAsValue</b>
<b>PivotAxis</b>	<b>View</b>
<b>PivotAxisMember</b>	<b>Axis</b> <b>ChildAxisMembers</b> <b>CustomGroupType</b> <b>FindAxisMember</b> <b>FindMember</b> <b>GroupField</b> <b>IsValid</b> <b>MemberProperties</b> <b>ParentAxisMember</b> <b>Path</b> <b>SourceMember</b> <b>TimeValue View</b>
<b>PivotCell</b>	<b>Data</b> <b>DetailColumnCount</b> <b>DetailRowCount</b>

	DetailTopOffset
	PageMember
PivotColumnMember	Axis ChildAxisMembers ChildColumnMembers CustomGroupType DetailLeftOffset DetailsExpanded FindAxisMember FindColumnMember FindMember GroupField IsValid MemberProperties ParentAxisMember ParentColumnMember Path SourceMember TimeValue TotalColumnMember View
PivotData	CellsEx Control CurrentCell IsConsistent Left2 LeftOffset PageAxis Top2 TopOffset View ViewportLeft ViewportTop
PivotDataAxis	View

<b>PivotDetailCell</b>	<b>Bookmark</b>
<b>PivotField</b>	<b>CustomGroupMembers</b>
	<b>ExcludedMembers</b>
	<b>Expression</b>
	<b>FilterContext</b>
	<b>FilterFunctionValue</b>
	<b>FilterOn</b>
	<b>FilterOnScope</b>
	<b>GroupedBackColor</b>
	<b>GroupedFont</b>
	<b>GroupedForeColor</b>
	<b>GroupedHAlignment</b>
	<b>GroupedHeight</b>
	<b>GroupEnd</b>
	<b>GroupStart</b>
	<b>IncludedMembers</b>
	<b>IsHyperlink</b>
	<b>MemberCaptions</b>
	<b>MemberProperties</b>
	<b>MemberPropertiesOrder</b>
	<b>OrderedMembers</b>
	<b>PropertyCaptionWidth</b>
	<b>PropertyHeight</b>
	<b>PropertyValueWidth</b>
	<b>SubtotalLabelBackColor</b>
	<b>SubtotalLabelFont</b>
	<b>SubtotalLabelForeColor</b>
	<b>SubtotalLabelHAlignment</b>
<b>PivotFieldSet</b>	<b>AllIncludeExclude</b>
	<b>AllMember</b>
	<b>AllowMultiFilter</b>
	<b>AlwaysIncludeInCube</b>
	<b>CompareMemberCaptionsBy</b>
	<b>CompareOrderedMembersBy</b>



	DefaultMember
	DisplayInFieldList
	FilterCaption
	FindMember
	Member View
PivotFilterAxis	View
PivotGroupAxis	View
PivotMember	CustomGroupType
	FindMember
	IsValid
	Path
	TimeValue
	View
PivotMemberProperty	DisplayIn
PivotPageMember	Axis
	ChildAxisMembers
	ChildPageMembers
	CustomGroupType
	FindAxisMember
	FindMember
	FindPageMember
	GroupField
	IsValid
	MemberProperties
	ParentAxisMember
	ParentPageMember
	Path
	SourceMember
	TimeValue
	TotalPageMember
	View
PivotResultAxis	Data
	SourceAxis
PivotResultColumnAxis	Data

	GroupFields Member SourceAxis SourceColumnAxis
PivotResultDataAxis	Data SourceAxis SourceDataAxis
PivotResultField	Axis SourceField
PivotResultFilterAxis	Data SourceAxis SourceFilterAxis
PivotResultGroupAxis	Data GroupFields Member SourceAxis
PivotResultGroupField	Axis DisplayTotal SourceField
PivotResultLabel	SourceLabel
PivotResultMemberProperty	MemberProperty
PivotResultPageAxis	Data GroupFields Member PageMember SourceAxis SourcePageAxis
PivotResultRowAxis	Data GroupFields Member SourceAxis SourceRowAxis

## PivotRowMember

Axis  
ChildAxisMembers  
ChildRowMembers  
CustomGroupType  
FindAxisMember  
FindMember  
FindRowMember  
GroupField  
IsValid  
MemberProperties  
ParentAxisMember  
ParentRowMember  
Path  
SourceMember  
TimeValue  
TotalRowHeight  
TotalRowMember  
View

---

## PivotTable

ActiveObject  
AllowCustomOrdering  
Commands  
DataSourceName  
DisplayDesignTimeUI  
DisplayOfficeLogo  
DisplayScreenTips  
EditMode  
Hwnd  
IsDirty  
Toolbar  
UserMode  
ViewOnlyMode

---

## PivotTotal

DisplayInFieldList  
Expression  
ShowAs

	SolveOrder
	View
PivotView	AllowAdditions AllowDeletions AllowEdits Control DisplayCalculatedMembers ExpandDetails ExpandMembers FilterCrossJoins HeaderBackColor HeaderFont HeaderForeColor HeaderHAlignment HeaderHeight PageAxis PropertyCaptionFont PropertyCaptionHAlignment PropertyValueFont PropertyValueHAlignment
Protection	AllowFormattingColumns AllowFormattingRows AllowHeadingRename Application
Range	Application CurrentArray End FormulaArray HasArray HorizontalAlignment MergeCells Next Offset

PrefixCharacter  
Previous  
ReadingOrder  
UseStandardHeight  
UseStandardWidth  
Value2  
VerticalAlignment  
Worksheet

SchemaField	IsHyperlink
SchemaParameter	IsHyperlink
Sheets	Application
Spreadsheet	ActiveWindow ActiveWorkbook Application Build Calculation CalculationVersion Commands DesignMode DisplayColumnHeadings DisplayDesignTimeUI DisplayOfficeLogo DisplayRowHeadings DisplayWorkbookTabs International LanguageSettings Names Sheets Toolbar ViewOnlyMode Windows Workbooks Worksheets XMLURL

<b>TitleBar</b>	<b>Application</b>
<b>Window</b>	<b>Application</b> <b>ColumnHeadings</b> <b>DisplayColumnHeadings</b> <b>DisplayCustomHeadings</b> <b>DisplayHeadings</b> <b>DisplayRowHeadings</b> <b>DisplayWorkbookTabs</b> <b>DisplayZeros</b> <b>EnableResize</b> <b>GridlineColor</b> <b>GridlineColorIndex</b> <b>RangeSelection</b> <b>RowHeadings</b> <b>ScrollColumn</b> <b>ScrollRow</b> <b>SelectedSheets</b> <b>TabRatio</b> <b>UsableHeight</b> <b>UsableWidth</b> <b>WindowNumber</b>
<b>Windows</b>	<b>Application</b>
<b>Workbook</b>	<b>Application</b> <b>CalculationVersion</b> <b>Colors</b> <b>Names</b> <b>ProtectStructure</b> <b>Sheets</b> <b>Windows</b> <b>Worksheets</b>
<b>Workbooks</b>	<b>Application</b>
<b>Worksheet</b>	<b>Application</b> <b>IsDataBound</b> <b>Names</b>

[Next](#)

[Previous](#)

[ProtectContents](#)

[ProtectionMode](#)

[StandardHeight](#)

[StandardWidth](#)

---

[Worksheets](#)



---

---

[Application](#)

---

# AllGroupingDefs Collection Object

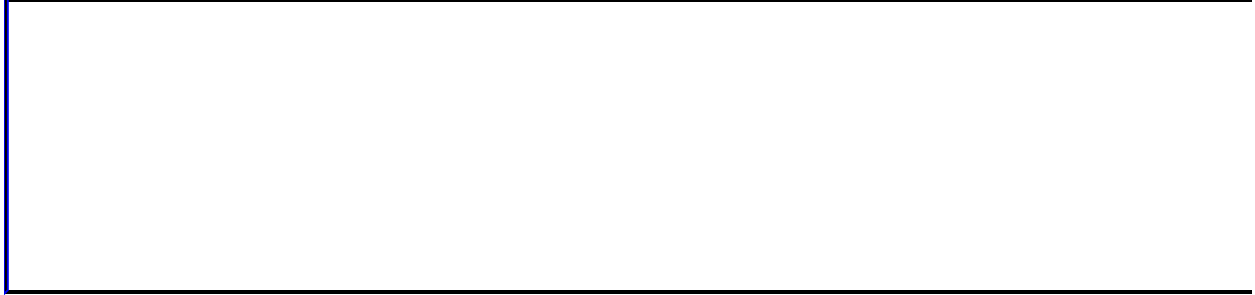
[DataSourceControl](#)  [AllGroupingDefs](#)  
 [GroupingDef](#)

Contains all of the [GroupingDef](#) objects in the data source control.



## Using the AllGroupingDefs Collection

The [DataSourceControl](#) object's [AllGroupingDefs](#) property returns an **AllGroupingDefs** collection.



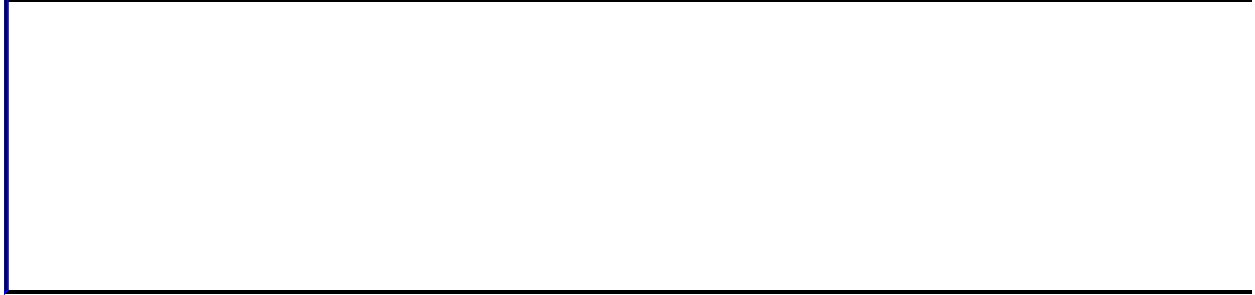
# AllPageFields Collection Object

[DataSourceControl](#) └ [AllPageFields](#)  
└ [PageField](#)

Contains all the [\*\*PageField\*\*](#) objects in the data source control.

## Using the AllPageFields Collection

The [DataSourceControl](#) object's [AllPageFields](#) property returns an **AllPageFields** collection.



# AutoFilter Object

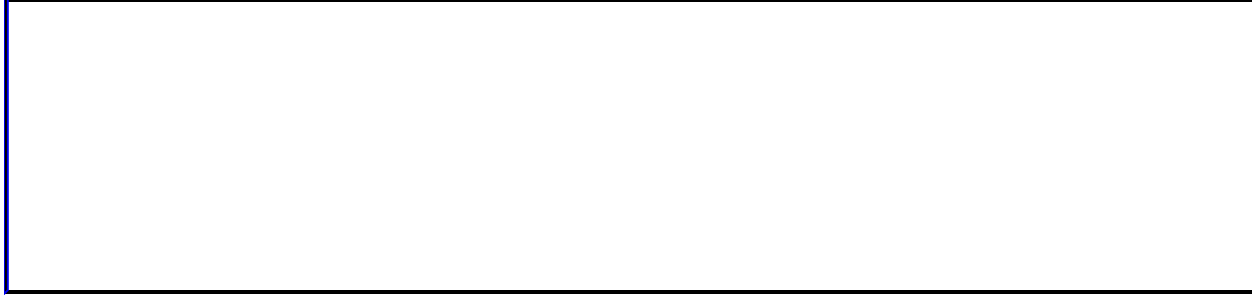
[Worksheet](#) └ [AutoFilter](#)

└ Multiple objects

Represents the AutoFilter container. The **AutoFilter** object contains a **Range** collection and a **Filters** collection.

## Using the AutoFilter Object

The [Worksheet](#) object's [AutoFilter](#) property returns the **AutoFilter** object for the specified worksheet.



# Border Object

[Borders](#) └ [Border](#)

Represents the border of an object.

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

## Using the Border Object

The [Borders](#) collection's [Item](#) property returns a **Border** object.



# Borders Collection Object

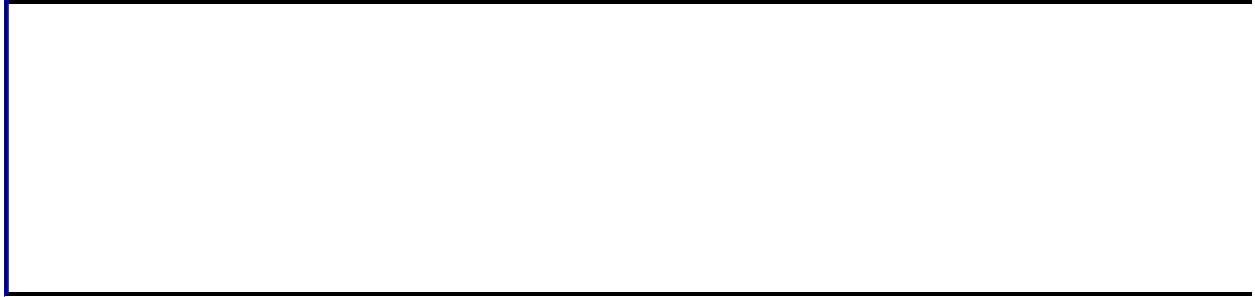
[Range](#) └ [Borders](#)  
└ [Border](#)

A collection of four [Border](#) objects that represent the four borders of a worksheet range.



## Using the Borders Collection

The [Range](#) object's [Borders](#) property returns a **Borders** collection.



# ByRef Object

[ByRef](#)

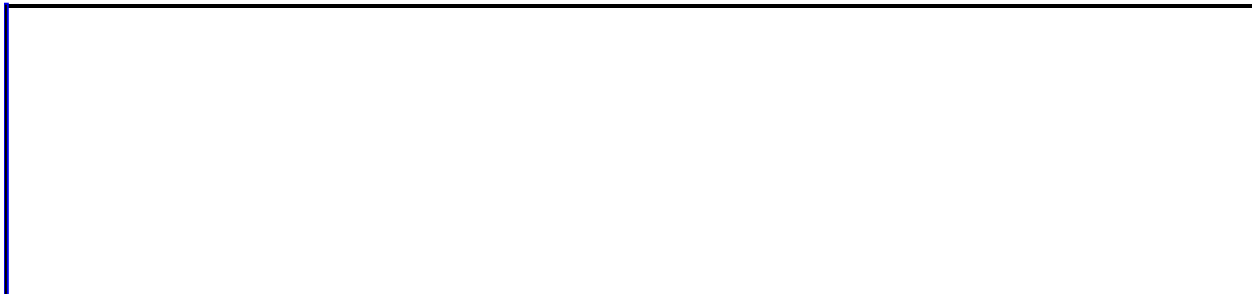
Contains the value of an event parameter.

## Using the ByRef object

Some events return or set values through parameters that are typed as **ByRef** objects. When an event parameter is typed as a **ByRef** object, use the [Value](#) property of the parameter to return or set the parameter.

The following event parameters return or set values through a **ByRef** object:

- The [BeforeContextMenu](#) event's *Menu* and *Cancel* parameters
- The [BeforeKeyDown](#) event's *Cancel* parameter
- The [BeforeKeyPress](#) event's *Cancel* parameter
- The [BeforeKeyUp](#) event's *Cancel* parameter
- The [BeforeRender](#) event's *Cancel* parameter
- The [CommandBeforeExecute](#) event's *Cancel* parameter
- The [CommandChecked](#) event's *Checked* parameter
- The [CommandEnabled](#) event's *Enabled* parameter
- The [CommandTipText](#) event's *Caption* parameter
- The [EndEdit](#) event's *FinalValue*, *Cancel*, and *ErrorDescription* parameters
- The [StartEdit](#) event's *InitialValue*, *Cancel*, and *ErrorDescription* parameters (Spreadsheet)
- The [StartEdit](#) event's *InitialValue*, *ArrowMode*, *CaretPosition*, *Cancel*, and *ErrorDescription* parameters (PivotTable)



# ChartSpace Object

[ChartSpace](#) └ Multiple objects

Represents the chart workspace. The chart workspace is the top-level chart container; it can contain more than one chart, with each chart represented by a [ChChart](#) object. When a chart workspace is first created, it is empty (it does not contain any charts). Use the [Add](#) method of the **ChCharts** object to create a new chart.

## Using the ChartSpace Object

You can use either the **CreateObject** method or the **New** keyword to create a new **ChartSpace** object.

The object ID for a chart control on an HTML page or a Visual Basic form returns a **ChartSpace** object.

The programmatic identifier for the **ChartSpace** object is CLSID:0002E556-0000-0000-C000-000000000046. The following example creates a chart workspace named "ChartSpace1" on an HTML page.

```
<object id=ChartSpace1 classid=CLSID:0002E556-0000-0000-C000-00000000
```



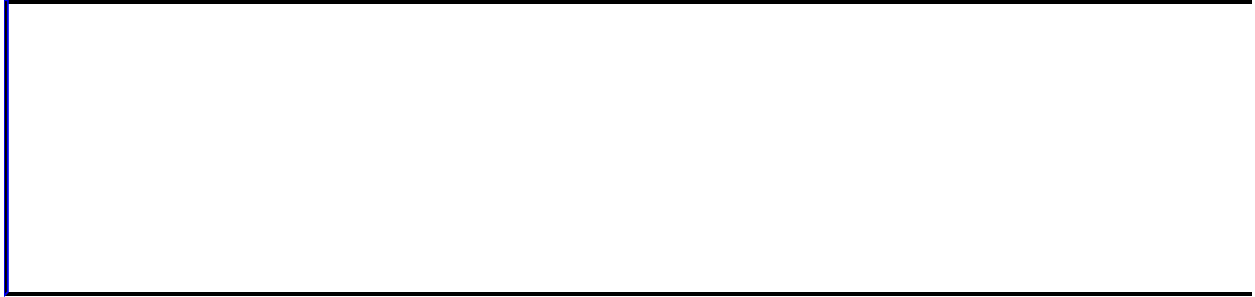
# ChAxes Collection

[ChChart](#) └ [ChAxes](#)  
└ [ChAxis](#)

The collection of **ChAxis** objects that represent the axes for a single chart. Each chart can have up to sixteen axes.

## Using the ChAxes collection

The **ChChart** object's [Axes](#) property returns a **ChAxes** collection.



# ChAxis Object

[ChAxes](#) └ [ChAxis](#)  
└ Multiple objects

Represents a single axis on a chart. A chart can have up to sixteen axes. The **ChAxis** object is a member of the **ChAxes** collection.



## Using the ChAxis object

Use the [ChAxes](#) object's [Add](#) method to add an axis to a chart.

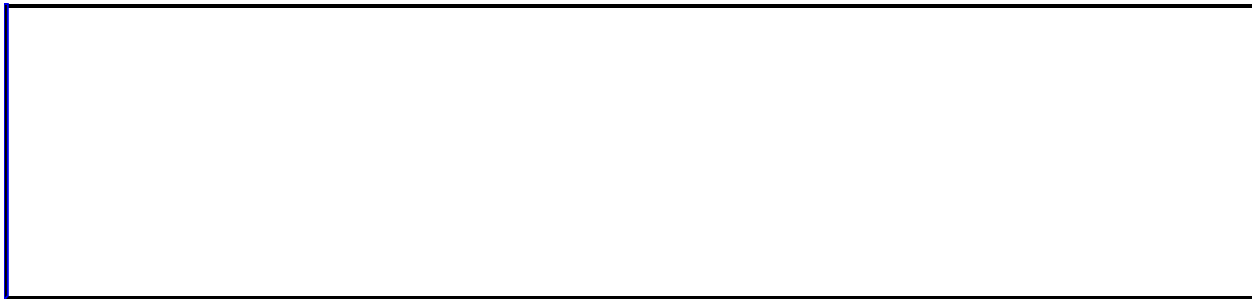
The following properties and methods return a **ChAxis** object.

The [ChAxes](#) object's [Add](#) method

The [ChAxes](#) object's [Item](#) property

The **ChAxis** object's [CrossingAxis](#) property

The [ChGridlines](#) object's [Parent](#) property



# ChBorder Object

Multiple objects  [ChBorder](#)

Represents the border of an object on a chart.

## Using the ChBorder object

The following properties return a **ChBorder** object:

The [ChChart](#) object's [Border](#) property

The [ChChartDraw](#) object's **Border** property

The [ChDataLabel](#) object's **Border** property

The [ChDataLabels](#) object's **Border** property

The [ChDropZone](#) object's [ButtonBorder](#) property

The [ChDropZone](#) object's [WatermarkBorder](#) property

The [ChLegend](#) object's **Border** property

The [ChPlotArea](#) object's **Border** property

The [ChPoint](#) object's **Border** property

The [ChSegmentBoundary](#) object's **Border** property

The [ChSeries](#) object's **Border** property

The [ChSurface](#) object's **Border** property

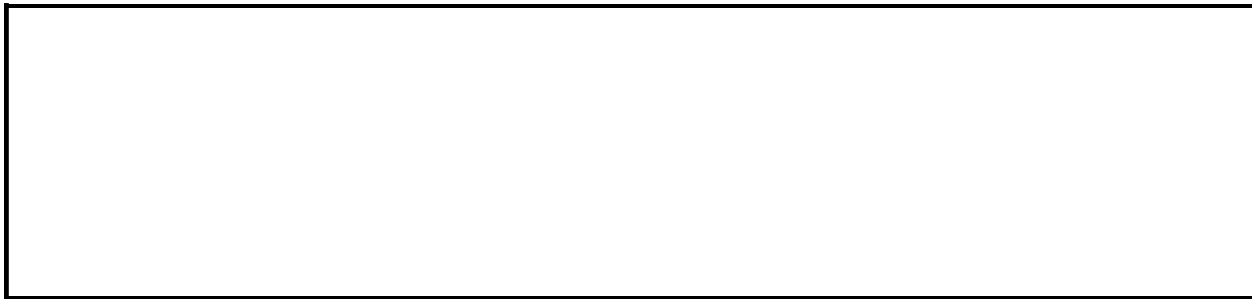
The [ChTitle](#) object's **Border** property

Use the [Color](#), [DashStyle](#), and [Weight](#) properties to set the attributes of a border. The following example sets border properties for the legend of ChartSpace1.

```
Sub Format_Chartspace_Legend()  
    Dim ChartLegend  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants
```

```
' Set a variable to the legend for the chartspace.  
Set ChartLegend = ChartSpace1.ChartSpaceLegend  
  
' Set the legend color.  
ChartLegend.Border.Color = "Blue"  
  
' Set the line weight for the legend.  
ChartLegend.Border.Weight = chConstants.owcLineWeightThick  
  
' Set the line style for the legend.  
ChartLegend.Border.DashStyle = chLineRoundDot
```

End Sub



# ChCategoryLabel Object

[ChCategoryLabels](#) └ [ChCategoryLabel](#)

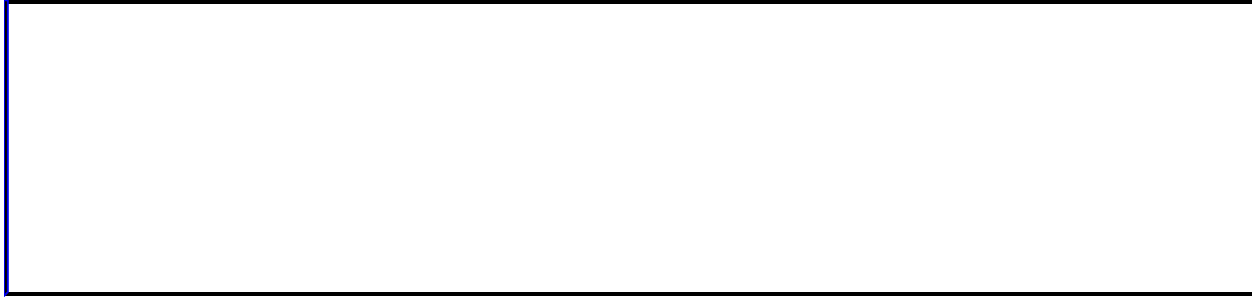
Represents a single label in a [ChCategoryLabels](#) collection.

## Using the ChCategoryLabel object

You can use the following properties to return a **ChCategoryLabel** object:

The [Item](#) property of the **ChCategoryLabels** collection

The [ParentLabel](#) property of the **ChCategoryLabel** object



# ChCategoryLabels Collection Object

[ChAxis](#) └ [ChCategoryLabels](#)  
└ Multiple objects

Represents a collection of all the [ChCategoryLabel](#) objects for the specified category axis.

## Using the ChCategoryLabels Collection Object

Use the [CategoryLabels](#) property of the [ChAxis](#) object to return a **ChCategoryLabels** collection. The following example displays the number of labels for the category axis on the first chart in Chartspace1.

```
Sub DisplayItemCount
```

```
    Dim chtChart1  
    Dim chConstants
```

```
    Set chConstants = Chartspace1.Constants
```

```
    Set chtChart1 = Chartspace1.Charts(0)
```

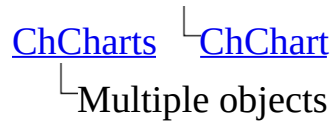
```
    MsgBox chtChart1.Axes(chConstants.chAxisPositionCategory) _  
        .CategoryLabels.ItemCount
```

```
End Sub
```





# ChChart Object



Represents a single chart in the chart workspace. The chart workspace can contain up to 16 charts. The **ChChart** object is a member of the [ChCharts](#) collection.

## Using the ChChart Object

Use the [ChCharts](#) collection object's [Add](#) method to add a chart to the chart workspace:

The following properties and methods return a **ChChart** object.

The [ChAxes](#) object's [Parent](#) property

The [ChAxis](#) object's **Parent** property

The [ChCharts](#) object's [Add](#) method

The **ChCharts** object's [Item](#) property

The [ChPlotArea](#) object's **Parent** property

The [ChSeries](#) object's **Parent** property

The [ChSeriesCollection](#) object's **Parent** property



# ChChartDraw Object

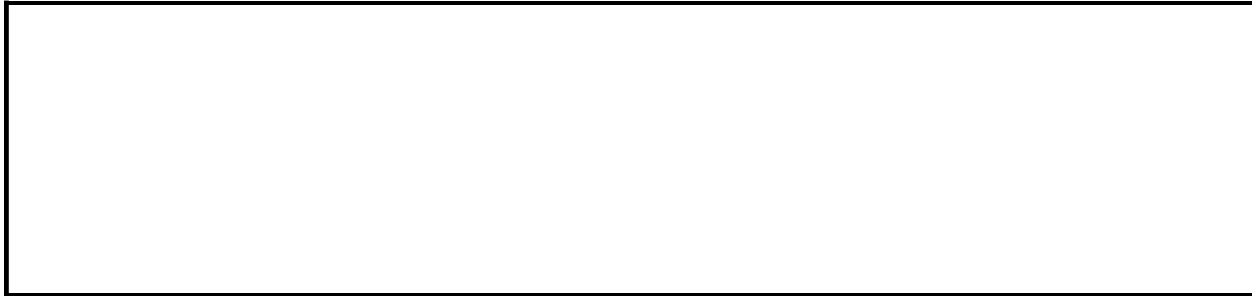
[ChChartDraw](#) └ Multiple objects

Can be used to draw items on a chart, such as a line, rectangle, or ellipse.

## Using the ChChartDraw object

The **DrawEllipse**, **DrawLine**, **DrawPolyLine**, **DrawPolygon**, **DrawLine** methods can be used to add drawing objects to a chart. The **DrawText** method can be used to add text to a chart. The **Border**, **Font**, **Interior**, and **Line** properties can be used to format each drawing object before it is added to the chart.

You must utilize one or more of the following events to add a drawing object to a chart: BeforeRender, AfterRender, or AfterFinalRender.



# ChChartField Object

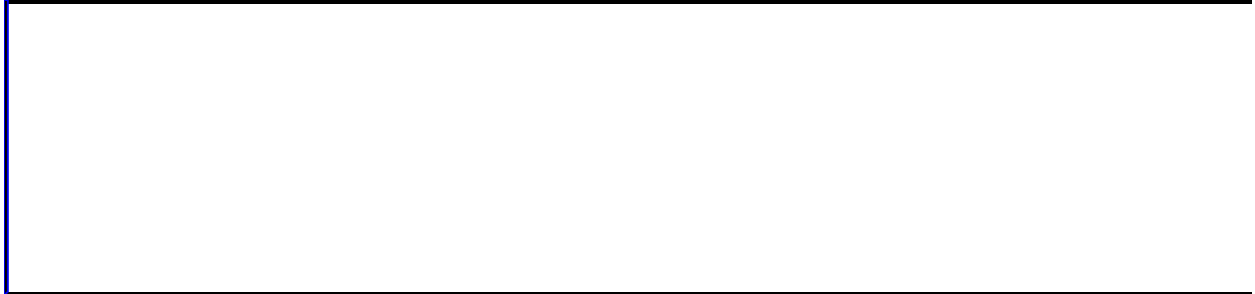
[ChChartFields](#) └ [ChChartField](#)

Represents a field in a drop zone.

## Using the ChChartField object

The [ChChartFields](#) object's **Item** property returns a **ChChartField** object.

Accessing this object when your chart is bound to literal data will result in a run-time error.



# ChChartFields Object

[ChDropZone](#) └ [ChChartFields](#)  
└ [ChChartField](#)

Represents the fields that have been added to a drop zone. Contains a collection of [ChChartField](#) objects.

## Using the ChChartFields object

The **ChDropZone** object's **ChartFields** property returns a **ChChartFields** object.





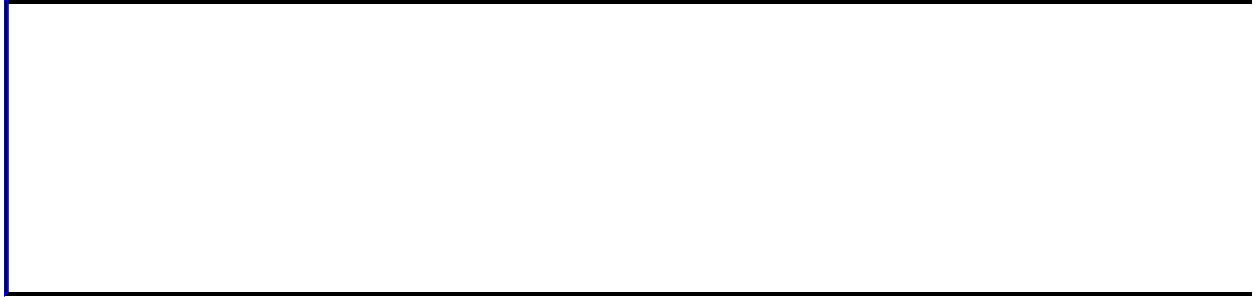
# ChCharts Collection

[ChartSpace](#) └ [ChCharts](#)  
└ [ChChart](#)



The collection of [ChChart](#) objects in the chart workspace. Each **ChChart** object represents a single chart. The chart workspace can contain up to 64 charts.

## Using the ChCharts collection

The [ChartSpace](#) object's [Charts](#) property returns a **ChCharts** collection.



# ChDataLabel Object

Multiple objects  [ChDataLabel](#)  
 Multiple objects

Represents a single data label for a series, or the single data label for a trendline.

## Using the ChDataLabel object

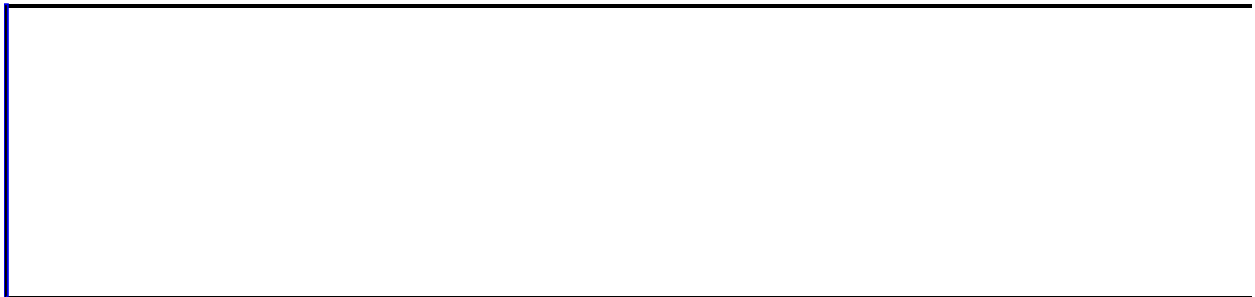
The following properties can be used to return a **ChDataLabel** object:

The [ChDataLabels](#) object's [Item](#) property

The [ChTrendline](#) object's [DataLabel](#) property

The following example adds data labels to the first series in the first chart in Chartspace1, and then formats the third data label.

```
Sub FormatSeriesLabel()  
  
    Dim serSeries1  
    Dim dlSeries1Labels  
  
    ' Set a variable to the first series of the first chart  
    ' in Chartspace1.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a set of data labels to the first series and return  
    ' a DataLabels object.  
    Set dlSeries1Labels = serSeries1.DataLabelsCollection.Add  
  
    dlSeries1Labels.Item(2).Font.Bold = True  
    dlSeries1Labels.Item(2).Font.Color = "Red"  
  
End Sub
```



# ChDataLabels Object

[ChDataLabelsCollection](#)  [ChDataLabels](#)

 Multiple objects

Contains a collection of [ChDataLabel](#) objects that represent all the data labels in the specified set of data labels for a series. Note that a series can contain more than one set of data labels.

## Using the ChDataLabels object

The following methods and properties can be used to return a ChDataLabels object:

The [ChDataLabelsCollection](#) collection object's [Add](#) method.

The **ChDataLabelsCollection** collection object's [Item](#) property.

The following example adds data labels to the first series in the first chart in Chartspace1 and then formats the data labels.

```
Sub AddDataLabels()  
  
    Dim serSeries1  
    Dim dlSeries1Labels  
  
    ' Set a variable to the first series of the first chart  
    ' in Chartspace1.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a set of data labels to the first series and return  
    ' a DataLabels object.  
    Set dlSeries1Labels = serSeries1.DataLabelsCollection.Add  
  
    ' Set the number format of the data labels.  
    dlSeries1Labels.NumberFormat = "0.00"  
  
    ' Set the data labels to display the category  
    ' name for the data point.  
    dlSeries1Labels.HasCategoryName = True  
  
    ' Set the data labels to display the value  
    ' for the data point.  
    dlSeries1Labels.HasValue = True  
  
End Sub
```



# ChDataLabelsCollection Collection

[ChSeries](#) └ [ChDataLabelsCollection](#)  
└ [ChDataLabels](#)

Represents the collection of [ChDataLabels](#) objects for a data series. Each **ChDataLabels** object represents a set of data labels for a data series.



## Using the ChDataLabelsCollection collection

Use the [DataLabelsCollection](#) property of the [ChSeries](#) object to return a **DataLabelsCollection** collection.

Use the [Add](#) method of the **ChDataLabelsCollection** collection to add a set of data labels to a data series.

The following example adds data labels to the first series in the first chart in Chartspace1, and then formats the data labels.

```
Sub AddDataLabels()  
  
    Dim serSeries1  
    Dim dlSeries1Labels  
  
    ' Set a variable to the first series of the first chart  
    ' in Chartspace1.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a set of data labels to the first series and return  
    ' a DataLabels object.  
    Set dlSeries1Labels = serSeries1.DataLabelsCollection.Add  
  
    ' Set the number format of the data labels.  
    dlSeries1Labels.NumberFormat = "0.00"  
  
    ' Set the data labels to display the category  
    ' name for the data point.  
    dlSeries1Labels.HasCategoryName = True  
  
    ' Set the data labels to display the value  
    ' for the data point.  
    dlSeries1Labels.HasValue = True  
  
End Sub
```



# ChDropZone Object

[ChartSpace](#) └ [ChDropZone](#)  
└ Multiple objects

Represents a drop zone on charts that are bound to a relational data source.

## Using the ChDropZone object

The [ChartSpace](#) object's [DropZones](#) method returns a **ChDropZone** object.

The following example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub Setup_DropZone()  
  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```



# ChErrorBars Object

[ChErrorBarsCollection](#) └ [ChErrorBars](#)  
└ [ChLine](#)

Represents the error bars for a series. Error bars indicate the degree of uncertainty for chart data. Only series in Radar, Polar, Area, Bar, Column, Line, and XY (Scatter) charts can have error bars. Only series in scatter charts can have x and y error bars. The **ChErrorBars** object is not a collection. There is 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 ChErrorBars object

Use the [Add](#) method of the [ChErrorBarsCollection](#) object to add error bars to a series.

The following methods return a **ChErrorBars** object. For more information, see the Help topics for these methods:

The [ChErrorBarsCollection](#) object's [Add](#) method

The [ChErrorBarsCollection](#) object's [Item](#) property

The following example adds error bars to the first series in the first chart in ChartSpace1, and then sets the properties for the error bars.

```
Sub AddErrorBars()  
  
    Dim chConstants  
    Dim ebCollection  
    Dim ebSeries1  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the collection of error bars for  
    ' the first series in the first chart of Chartspace1.  
    Set ebCollection = ChartSpace1.Charts(0).SeriesCollection(0).Err  
  
    ' Add error bars to the chart.  
    ebCollection.Add  
  
    ' Set a variable to the error bars for the data series.  
    Set ebSeries1 = ebCollection.Item(0)  
  
    ' Set the error bars so that they represent a certain  
    ' percentage of the value of a data point.  
    ebSeries1.Type = chConstants.chErrorBarTypePercent  
  
    ' The error bars represent 5% of a data point.  
    ebSeries1.Amount = 0.05  
  
End Sub
```



# ChErrorBarsCollection Object

[ChSeries](#) └ [ChErrorBarsCollection](#)  
└ [ChErrorBars](#)

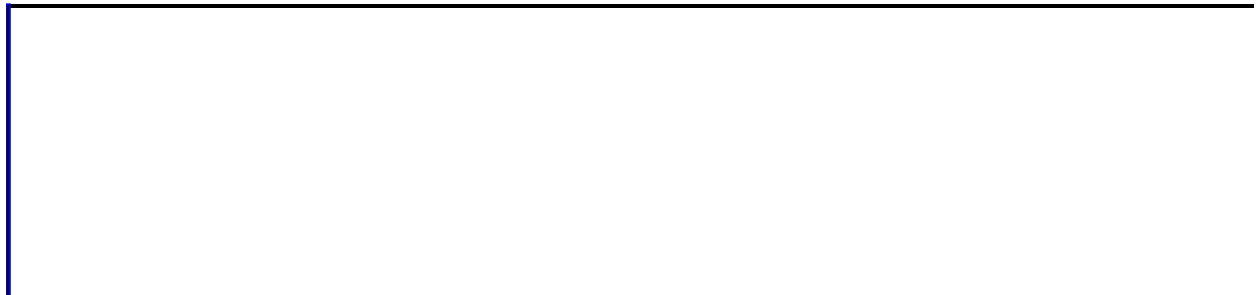
The collection of [ChErrorBars](#) objects for a single series.

## Using the ChErrorBarsCollection object

The [ChSeries](#) object's [ErrorBarsCollection](#) property returns a **ChErrorBarsCollection** object.

The following example adds error bars to the first series in the first chart in ChartSpace1, then sets the properties for the error bars.

```
Sub AddErrorBars()  
  
    Dim chConstants  
    Dim ebCollection  
    Dim ebSeries1  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the collection of error bars for  
    ' the first series in the first chart of Chartspace1.  
    Set ebCollection = ChartSpace1.Charts(0).SeriesCollection(0).ErrorBarsCollection  
  
    ' Add error bars to the chart.  
    ebCollection.Add  
  
    ' Set a variable to the error bars for the data series.  
    Set ebSeries1 = ebCollection.Item(0)  
  
    ' Set the error bars so that they represent a certain  
    ' percentage of the value of a data point.  
    ebSeries1.Type = chConstants.chErrorBarTypePercent  
  
    ' The error bars represent 5% of a data point.  
    ebSeries1.Amount = 0.05  
  
End Sub
```





# ChFont Object

Multiple objects  [ChFont](#)

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

## Using the ChFont object

The following properties can be used to return a **ChFont** object:

The [ChAxis](#) object's [Font](#) property

The [ChChartDraw](#) object's **Font** property

The [ChDataLabel](#) object's **Font** property

The **ChDataLabels** object's **Font** property

The [ChDropZone](#) object's [ButtonFont](#) property

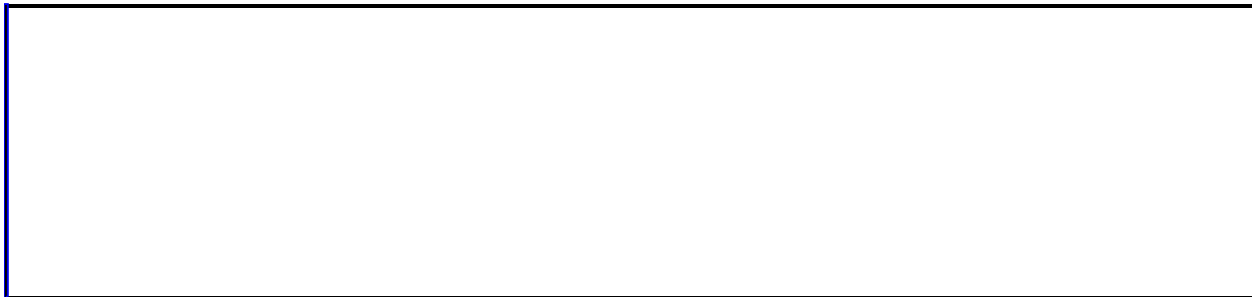
The **ChDropZone** object's [WatermarkFont](#) property

The [ChLegend](#) object's **Font** property

The [ChLegendEntry](#) object's **Font** property

The [ChTitle](#) object's **Font** property

Use the **Name** property to set the font for a particular object. The **Bold**, **Italic**, **Color**, **Underline**, and **Size** properties can be used to further format the font of a particular object.



# ChFormatMap Object

[ChSeries](#) └ [ChFormatMap](#)  
└ [ChSegments](#)

The **ChFormatMap** object allows formatting to represent a range of data values. The ChFormatMap object can be used provide visual cues that highlight certain portions of your data.

## Using the ChFormatMap object

The [FormatMap](#) property of the [ChSeries](#) object returns a **ChFormatMap** object.

Format maps contain one or more [ChSegment](#) objects, each of which can be formatted independently.

The following example binds ChartSpace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created. The smaller values are displayed in white, then larger values are displayed in a light shade of blue, and finally the largest values in the chart are displayed in dark blue.

```
Sub Window_Onload()  
  
    Dim serSeries1  
    Dim segSegment1  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind ChartSpace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=ServerName  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind ChartSpace1 to the Quantity  
    ' in the Order details table.  
    ChartSpace1.SetData chConstants.chDimCategories, chConstants.chDimValues  
    ChartSpace1.SetData chConstants.chDimValues, chConstants.chDimFormatValues  
  
    ' Create a format map.  
    ChartSpace1.SetData chConstants.chDimFormatValues, chConstants.chDimFormatValues  
  
    ' Set a variable to the first series in the first chart in ChartSpace1  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a segment to the format map.  
    Set segSegment1 = serSeries1.FormatMap.Segments.Add  
  
    ' Specify that the divisions in formatting be created automatically  
    segSegment1.HasAutoDivisions = True
```

```

' Measure the segment boundaries based upon a percentage.
segSegment1.Begin.ValueType = chConstants.chBoundaryValuePercent
segSegment1.End.ValueType = chConstants.chBoundaryValuePercent

' Set the beginning value to 0%, and the ending value to 100%.
segSegment1.Begin.Value = 0
segSegment1.End.Value = 1

' Format the interior of the matching values.
segSegment1.Begin.Interior.Color = "White"
segSegment1.End.Interior.Color = "Blue"

```

End Sub

The following example binds ChartSpace1 to the Order Details table in the Northwind database. Then, two segments are created. The first segment highlights the lowest 10% of values in the first series in the chart. The second segment highlights the top 20% of values in the first series in the chart.

```

Sub Window_Onload()

    Dim serseries1
    Dim segBottom10Pct
    Dim segTop20Pct
    Dim chConstants

    Set chConstants = ChartSpace1.Constants

    ' The following two lines of code bind ChartSpace1 to the Order D
    ' Northwind SQL Server database.
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Secur
                                "Catalog=Northwind;Data Source=Ser
    ChartSpace1.DataMember = "Order Details"

    ' The following two lines of code bind ChartSpace1 to the Quantit
    ' in the Order details table.
    ChartSpace1.SetData chConstants.chDimCategories, chConstants.chDa
    ChartSpace1.SetData chConstants.chDimValues, chConstants.chDataBo

    ' Create a format map.
    ChartSpace1.SetData chConstants.chDimFormatValues, chConstants.ch

    ' Set a variable to the first series in the first chart in Charts
    Set serseries1 = ChartSpace1.Charts(0).SeriesCollection(0)

    ' Add a segment to the format map. This segment will
    ' represent the bottom 10% of values in the chart.
    Set segBottom10Pct = serseries1.FormatMap.Segments.Add

```

```

' Measure the segment boundaries based upon a percentage.
segBottom10Pct.Begin.ValueType = chConstants.chBoundaryValuePerce
segBottom10Pct.End.ValueType = chConstants.chBoundaryValuePercent

' Set the beginning value to 0%, and the ending value to 10%.

segBottom10Pct.Begin.Value = 0

segBottom10Pct.End.Value = 0.1

' Format the interior of the matching values.
segBottom10Pct.Begin.Interior.Color = "red"
segBottom10Pct.End.Interior.Color = "red"

' Add a segment to the format map. This segment will
' represent the top 20% of values in the chart.
Set segTop20Pct = serseries1.FormatMap.Segments.Add

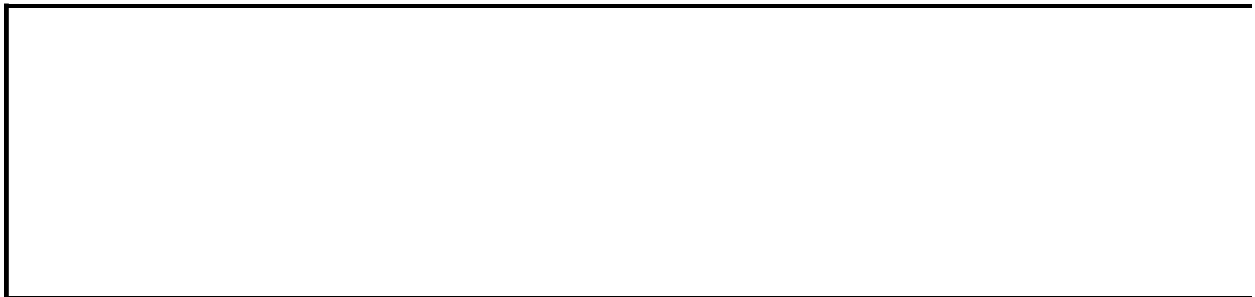
' Measure the segment boundaries based upon a percentage.
segTop20Pct.Begin.ValueType = chConstants.chBoundaryValuePercent
segTop20Pct.End.ValueType = chConstants.chBoundaryValuePercent

' Set the beginning value to 80%, and the ending value to 100%.
segTop20Pct.Begin.Value = 0.8
segTop20Pct.End.Value = 1

' Format the interior of the matching values.
segTop20Pct.Begin.Interior.Color = "green"
segTop20Pct.End.Interior.Color = "green"

```

End Sub



# ChGridlines Object

[ChAxis](#) └ [ChGridlines](#)  
└ [ChLine](#)

Represents major or minor gridlines on a chart axis. You cannot have gridlines without an 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 is not a collection. There is 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 ChGridlines object

The following properties return a **ChGridlines** object.

The [ChAxis](#) object's [MajorGridlines](#) property

The **ChAxis** object's [MinorGridlines](#) property

The following example enables the major and minor gridlines for the value axis in the first chart in ChartSpace1. Then, the weight of the gridlines is formatted.

```
Sub EnableGridlines()  
  
    Dim chConstants  
    Dim axValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the value axis in the first chart in Chartsp  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
  
    ' The following two lines of code turn on the major and  
    ' minor gridlines for the value axis.  
    axValueAxis.HasMajorGridlines = True  
    axValueAxis.HasMinorGridlines = True  
  
    ' The following two lines of code set the line weight of the  
    ' major and minor gridlines for the value axis.  
    axValueAxis.MajorGridlines.Line.Weight = chConstants.owcLineWeig  
    axValueAxis.MinorGridlines.Line.Weight = chConstants.owcLineWeig  
  
End Sub
```





# ChInterior Object

Multiple objects  [ChInterior](#)

Represents the interior formatting of an object.

## Using the ChInterior object

The following properties return a **ChInterior** object:

The [ChChart](#) object's [Interior](#) property

The [ChChartDraw](#) object's **Interior** property

The [ChartSpace](#) object's **Interior** property

The [ChDataLabel](#) object's **Interior** property

The [ChDataLabels](#) object's **Interior** property

The [ChDropZone](#) object's [ButtonInterior](#) property

The [ChDropZone](#) object's [WatermarkInterior](#) property

The [ChLegend](#) object's **Interior** property

The [ChPlotArea](#) object's **Interior** property

The [ChPoint](#) object's **Interior** property

The [ChSegmentBoundary](#) object's **Interior** property

The [ChSeries](#) object's **Interior** property

The [ChSurface](#) object's **Interior** property

The [ChTitle](#) object's **Interior** property

The following example sets the interior fill of the first two series and the plot area of the first chart in ChartSpace1.

```
Sub FormatInteriorFills()
```

```
    Dim chConstants  
    Dim serSeries1  
    Dim serSeries2
```

```
Set chConstants = ChartSpace1.Constants

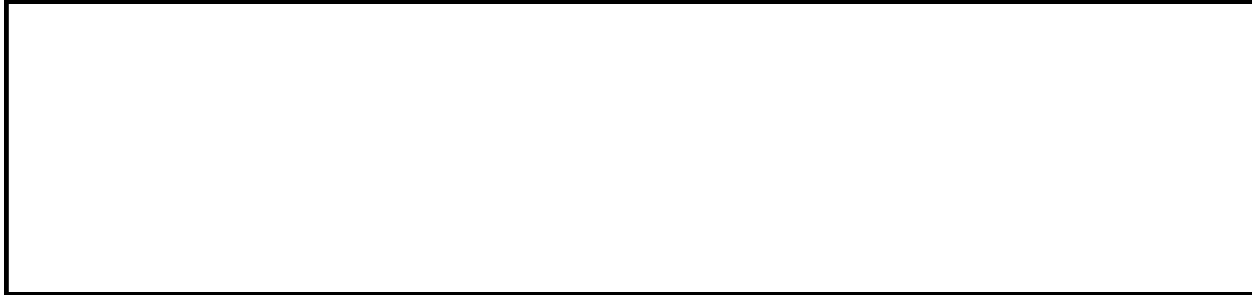
Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)
Set serSeries2 = ChartSpace1.Charts(0).SeriesCollection(1)

' Set the interior fill of the first series to a two-color gradi
serSeries1.Interior.SetTwoColorGradient chConstants.chGradientDi
    chConstants.chGradientVariantCenter, "Blue", "Silver"



' Set the interior fill of the second series to a solid color.
serSeries2.Interior.SetSolid "Purple"

' Set the interior fill of the plot area to a preset texture.
ChartSpace1.Charts(0).PlotArea.Interior.SetTextured _
    chConstants.chTextureParchment, chConstants.chTile
```

End Sub



# ChLegend Object

Multiple objects  [ChLegend](#)  
 Multiple objects

Represents a chart workspace or chart legend. A chart or chart workspace can have only one legend. The **ChLegend** object contains a [ChLegendEntries](#) collection that contains one or more [ChLegendEntry](#) objects.

## Using the ChLegend object

Use the [HasLegend](#) property or the [HasChartSpaceLegend](#) property to enable the legend.

The following properties return a **ChLegend** object.

The [ChartSpace](#) object's [ChartSpaceLegend](#) property

The [ChChart](#) object's [Legend](#) property

The [ChLegendEntries](#) object's [Parent](#) property

The [ChLegendEntry](#) object's **Parent** property



# ChLegendEntries Collection

[ChLegend](#) └ [ChLegendEntries](#)  
└ [ChLegendEntry](#)

The collection of [ChLegendEntry](#) objects for the specified legend.

## Using the ChLegendEntries collection

The [ChLegend](#) object's [LegendEntries](#) property returns a **ChLegendEntries** collection.

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(0) is at the top of the legend, and LegendEntries(LegendEntries.Count - 1) is at the bottom. The following example changes the font for the text of the legend entry at the top of the chartspace legend (this is usually the legend for series one) in Chartspace1.

```
Chartspace1.ChartSpaceLegend.LegendEntries(0) _  
    .Font.Bold = True
```



# ChLegendEntry Object

[ChLegendEntries](#) └ [ChLegendEntry](#)  
└ [ChFont](#)

Represents a single legend entry. The **ChLegendEntry** object is a member of the **ChLegendEntries** collection.



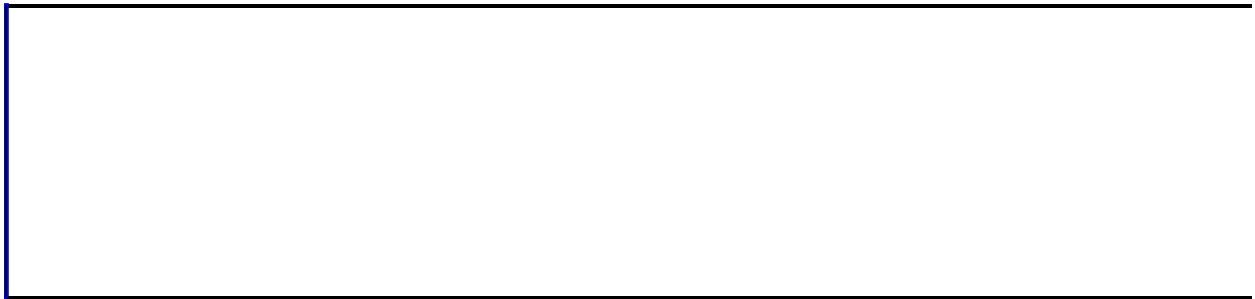
## Using the ChLegendEntry object

The [ChLegendEntries](#) collection's [Item](#) property returns a **ChLegendEntry** 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(0) 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 chartspace legend (this is usually the legend for series one) in Chartspace1.

```
Chartspace1.ChartSpaceLegend.LegendEntries(0) _  
    .Font.Bold = True
```



# ChLine Object

Multiple objects  [ChLine](#)

Represents the formatting of a line on a chart.

## Using the ChLine object

You can use the following properties to return a ChLine object:

The [ChAxis](#) object's **Line** property

The [ChChartDraw](#) object's **Line** property

The [ChErrorBars](#) object's **Line** property

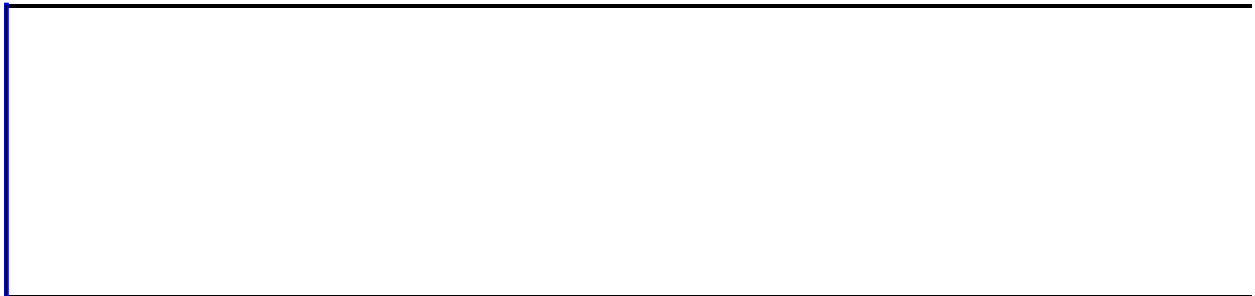
The [ChGridlines](#) object's **Line** property

The [ChPoint](#) object's **Line** property

The [ChSegmentBoundary](#) object's **Line** property

The [ChSeries](#) object's **Line** property

The [ChTrendline](#) object's **Line** property



# ChMarker Object

[ChSeries](#) └ [ChMarker](#)

Represents a data marker on a Line, XY (Scatter), Radar, or Polar chart.

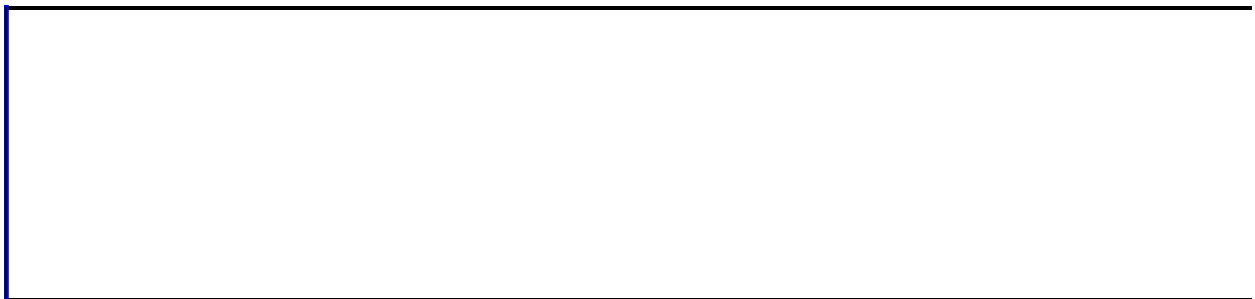
## Using the ChMarker object

The [ChSeries](#) object's [Marker](#) property returns a **ChMarker** object.

Use the **Size** and **Style** properties to format **ChMarker** objects.

The following example converts the first series in the first chart of Chartspace1 to a line chart, and then formats the markers on the line.

```
Sub FormatMarkers()  
  
    Dim serSeries1  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the first series of the first chart  
    ' in Chartspace1.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Change the type of the first data series to a line chart.  
    serSeries1.Type = chConstants.chChartTypeLineMarkers  
  
    ' Set the marker style.  
    serSeries1.Marker.Style = chConstants.chMarkerStyleDiamond  
  
    ' Set the size of the markers.  
    serSeries1.Marker.Size = 7  
  
End Sub
```



# ChPlotArea Object

[ChChart](#) └ [ChPlotArea](#)  
└ Multiple objects

Represents the plot area on a chart (the area where the chart data is plotted). Pie, Doughnut, Radar, and Polar charts do not have a plot area; instead, these charts draw directly on the chart area.

## Using the ChPlotArea object

The [ChChart](#) object's [PlotArea](#) property returns a **ChPlotArea** object.

The following example fills the plot area of the first chart in Chartspace1 with a predefined texture.

```
Sub FormatPlotArea()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set the interior fill of the plot area to a preset texture.  
    ChartSpace1.Charts(0).PlotArea.Interior.SetTextured _  
  
    chConstants.chTextureParchment, chConstants.chTile  
  
End Sub
```



# ChPoint Object

[ChPoints](#) └ [ChPoint](#)  
└ Multiple objects

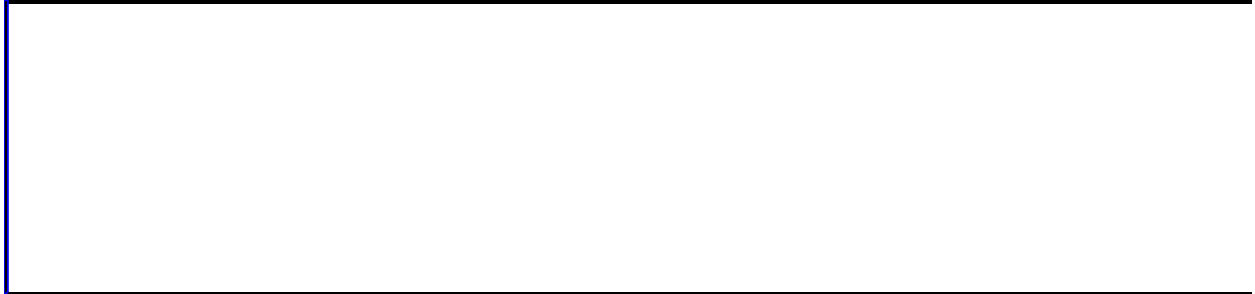
Represents a single data point in a series on a chart. The **ChPoint** object is a member of the [ChPoints](#) collection, which contains all the points in a given series.



## Using the ChPoint object

Use this object to format single data points in a series, or use the [GetValue](#) method to return a point value.

The **ChPoints** object's [Item](#) property returns a **ChPoint** object.



# ChPoints Collection

[ChSeries](#) └ [ChPoints](#)  
└ [ChPoint](#)

A collection of all the [ChPoint](#) objects in a data series. The number of data points in a series is limited only by the amount of system memory in the computer being used.

## Using the ChPoints collection

The [ChSeries](#) object's [Points](#) property returns a **ChPoints** collection.

Use **Points(index)**, where *index* is the point index number, to return a single **ChPoint** object. Points are numbered from left to right on the series. **Points(0)** is the leftmost point, and **Points(Points.Count - 1)** is the rightmost point. The following example stores the value of the third point in the first data series of the first chart in Chartspace1 in a variable.

```
Sub GetPointValue()  
  
    Dim ptSeries1Points  
    Dim dblPointValue  
  
    ' Set a variable to the collection of points for the first  
    ' data series in the first chart in Chartspace1.  
    Set ptSeries1Points =  
ChartSpace1.Charts(0).SeriesCollection(0).Points  
  
    ' Store the underlying value of the third data point in a  
    variable.  
    dblPointValue = ptSeries1Points(2).GetValue(chDimValues)  
  
End Sub
```



# ChScaling Object

Multiple objects  [ChScaling](#)

Represents the scaling for a data series, axis, or chart.

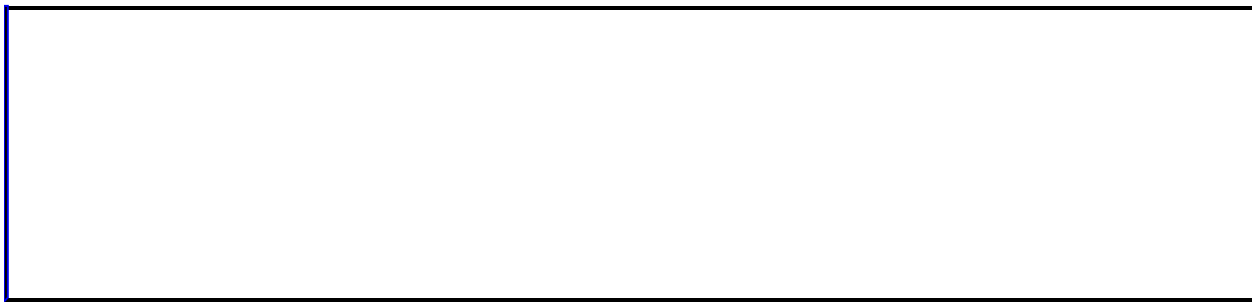
## Using the ChScaling object

The following properties return a **ChScaling** object:

The [ChAxis](#) object's [Scaling](#) property

The [ChChart](#) object's [Scalings](#) property

The [ChSeries](#) object's [Scalings](#) property



# ChSegment Object

[ChSegments](#) └ [ChSegment](#)  
└ [ChSegmentBoundary](#)

Represents a single segment in a format map. Each segment of a format map can be formatted independently of the other segments.

## Using the ChSegment object

The following methods and properties return a **ChSegment** object.

The [ChSegments](#) object's [Add](#) method

The **ChSegments** object's [Item](#) property

Use the **Add** method of the **ChSegments** object to create a new segment. Use the properties of the **ChSegmentBoundary** object returned by the **Begin** property to format the beginning of a segment. Use the properties of the **ChSegmentBoundary** object returned by the **End** property to format the end of a segment.

## Example

The following example binds ChartSpace1 to the Order Details table in the Northwind database. Then, two segments are created. The first segment highlights the lowest 10% of values in the first series in the chart. The second segment highlights the top 20% of values in the first series in the chart.

```
Sub Window_Onload()  
  
    Dim serseries1  
    Dim segBottom10Pct  
    Dim segTop20Pct  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind ChartSpace1 to the Order  
    ' Northwind SQL Server database.  
  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "User ID=sa;Initial Catalog=Northwind;  
                                   "ServerName=.;PASSWORD=;"  
  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind ChartSpace1 to the Quantity  
    ' in the Order details table.  
    ChartSpace1.SetData chConstants.chDimCategories, chConstants.chDimValues  
    ChartSpace1.SetData chConstants.chDimValues, chConstants.chDimFormatValues  
  
    ' Create a format map.  
    ChartSpace1.SetData chConstants.chDimFormatValues, chConstants.chDimValues  
  
    ' Set a variable to the first series in the first chart in ChartSpace1  
    Set serseries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a segment to the format map. This segment will  
    ' represent the bottom 10% of values in the chart.  
    Set segBottom10Pct = serseries1.FormatMap.Segments.Add  
  
    ' Measure the segment boundaries based upon a percentage.  
  
    segBottom10Pct.Begin.ValueType = chConstants.chBoundaryValuePercentage  
    segBottom10Pct.End.ValueType = chConstants.chBoundaryValuePercentage  
  
    ' Set the beginning value to 0%, and the ending value to 10%
```



```
segBottom10Pct.Begin.Value = 0
segBottom10Pct.End.Value = 0.1

' Format the interior of the matching values.
segBottom10Pct.Begin.Interior.Color = "red"
segBottom10Pct.End.Interior.Color = "red"

' Add a segment to the format map. This segment will
' represent the top 20% of values in the chart.

Set segTop20Pct = serseries1.FormatMap.Segments.Add

' Measure the segment boundaries based upon a percentage.
segTop20Pct.Begin.ValueType = chConstants.chBoundaryValuePercent
segTop20Pct.End.ValueType = chConstants.chBoundaryValuePercent

' Set the beginning value to 80%, and the ending value to 100%.
segTop20Pct.Begin.Value = 0.8
segTop20Pct.End.Value = 1

' Format the interior of the matching values.
segTop20Pct.Begin.Interior.Color = "green"
segTop20Pct.End.Interior.Color = "green"
```

End Sub



# ChSegmentBoundary Object

[ChSegment](#) └ [ChSegmentBoundary](#)  
└ Multiple objects

Represents the boundaries of of a [ChSegment](#) object.

## Using the ChSegmentBoundary object

The following properties return a **ChSegmentBoundary** object.

The [ChSegment](#) object's [Begin](#) property

The **ChSegment** object's [End](#) property

Use the [Value](#) property to set the beginning and ending values of a **ChSegmentBoundary** object. Use the [ValueType](#) property to specify whether a value represents a percentage or an absolute value.

Use the objects returned by the following properties to format a **ChSegmentBoundary** object: [Border](#), [Interior](#), and [Line](#).



# ChSegments Object

[ChFormatMap](#) └ [ChSegments](#)  
└ [ChSegment](#)

Represents the collection of segments for a [ChFormatMap](#) object.

## Using the ChSegments object

The [Segments](#) property of the **ChFormatMap** object can be used to return a **ChSegments** object.

Use the [Add](#) method of the **ChSegments** object to add a segment to a format map.

Use the [Item](#) property of the **ChSegments** object to return a single [ChSegment](#) object.



# ChSeries Object

[ChSeriesCollection](#) └ [ChSeries](#)  
└ Multiple objects

Represents a series on a chart. The **ChSeries** object is a member of the [ChSeriesCollection](#) collection.

## Using the ChSeries object

Use the [Add](#) method of the [ChSeriesCollection](#) collection to add a series to a chart.

The following properties and methods return a **ChSeries** object:

The [ChDataLabels](#) object's [Parent](#) property

The [ChDataLabelsCollection](#) object's **Parent** property

The [ChErrorBars](#) object's **Parent** property

The [ChErrorBarsCollection](#) object's **Parent** property

The [ChPoint](#) object's **Parent** property

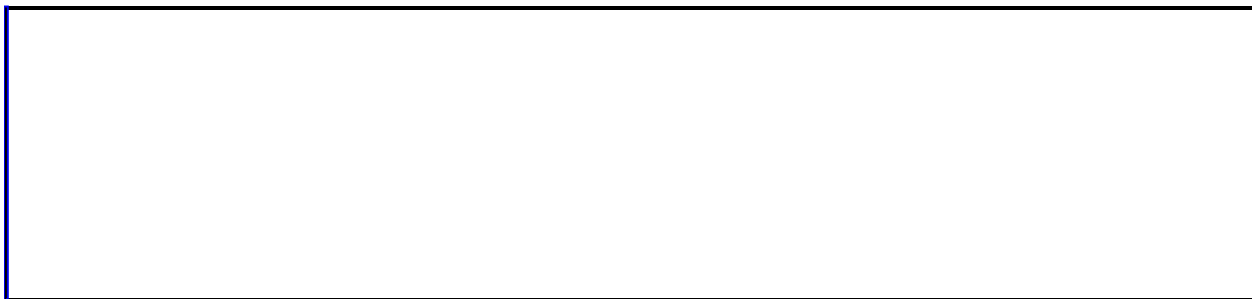
The [ChPoints](#) object's **Parent** property

The [ChSeriesCollection](#) object's [Add](#) method

The **ChSeriesCollection** object's [Item](#) property

The [ChTrendline](#) object's **Parent** property

The [ChTrendlines](#) object's **Parent** property



# ChSeriesCollection Collection Object

[ChChart](#) └ [ChSeriesCollection](#)

└ Multiple objects

A collection of all the [ChSeries](#) objects on a chart. A chart can contain up to 256 series.



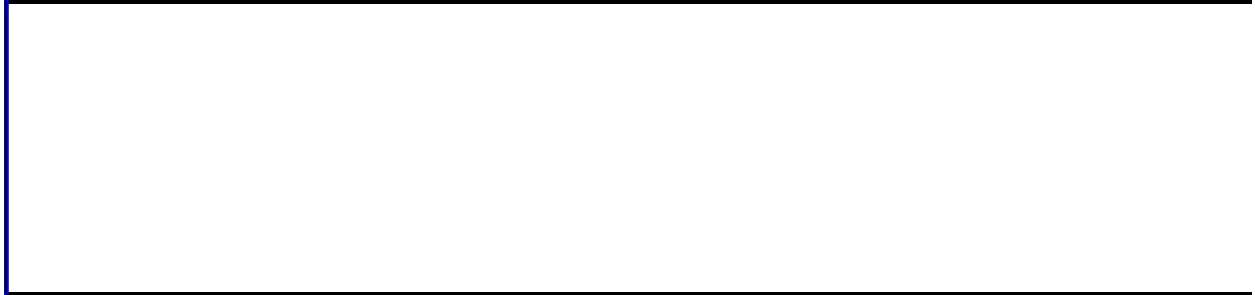
## Using the ChSeriesCollection Collection Object

The [ChChart](#) object's [SeriesCollection](#) property returns a **ChSeriesCollection** collection.

Use the **Add** method to create a new series and add it to the chart.

Use **SeriesCollection(index)**, where *index* is the series index number or name, to return a single **ChSeries** object. The following example sets the color of the interior for the first series in the first chart of ChartSpace1.

```
ChartSpace1.Charts(0).SeriesCollection(1).Interior.Color = "Red"
```



# ChSurface Object

[ChPlotArea](#) └ [ChSurface](#)  
└ Multiple objects

Represents the surface of the walls and floor of a chart.

## Using the ChSurface object

The following properties can be used to return a **ChSurface** object:

The [ChChart](#) object's [BackWall](#) property



The **ChChart** object's [SideWall](#) property

The **ChChart** object's [Floor](#) property

You can use the **Border**, **Interior**, and **Thickness** properties to format a **ChSurface** object.



# ChTitle Object

Multiple objects  [ChTitle](#)  
 Multiple objects

Represents the title of a chart workspace, axis, or chart.

## Using the ChTitle object

Use the [HasTitle](#) or [HasChartspaceTitle](#) property to enable titles.

The following properties return a **ChTitle** object:

The [ChartSpace](#) object's [ChartSpaceTitle](#) property

The [ChAxis](#) object's [Title](#) property

The [ChChart](#) object's **Title** property

The following example adds a title to the first chart in Chartspace1 and then formats the newly-created title.

```
Sub AddChartTitle()  
    Dim Chart1Title  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Enable the title for the first chart in the  
    ' chartspace.  
    ChartSpace1.Charts(0).HasTitle = True  
  
    ' Set a variable to the chart title.  
    Set Chart1Title = ChartSpace1.Charts(0).Title  
  
    ' Set the caption for the title.  
    Chart1Title.Caption = "2000 Sales by Department"  
  
    ' Set the title to display at the bottom of the chart.  
    Chart1Title.Position = chConstants.chTitlePositionBottom  
  
    ' Format the font used for the title.  
    Chart1Title.Font.Bold = True  
    Chart1Title.Font.Name = "Tahoma"  
    Chart1Title.Font.Size = 16  
  
End Sub
```



# ChTrendline Object

[ChTrendlines](#) └ [ChTrendline](#)  
└ Multiple objects

Represents a trendline on a chart. A trendline shows the trend, or direction, of data in a series. The **ChTrendline** object is a member of the [ChTrendlines](#) collection.

## Using the ChTrendline object

Use the [Add](#) method of the [ChTrendlines](#) object to add a trendline to a series.

The following method and property return a **ChTrendline** object.

The **ChTrendlines** collection's [Add](#) method

The **ChTrendlines** collection's [Item](#) property

The following example adds a trendline to the first series in the first chart in Chartspace1 and then formats the trendline.

```
Sub AddPolyTrendline()  
  
    Dim serSeries1  
    Dim chConstants  
    Dim tlSeries1Trend  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the first series of the first chart  
    ' in Chartspace1.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a trendline to the first series and return  
    ' a Trendline object.  
    Set tlSeries1Trend = serSeries1.Trendlines.Add  
  
    ' Display the equation used to calculate the trendline.  
    tlSeries1Trend.IsDisplayingEquation = True  
  
    ' Set the trendline to be a polynomial trendline.  
    tlSeries1Trend.Type = chConstants.chTrendlineTypePolynomial  
  
End Sub
```





# ChTrendlines Collection

[ChSeries](#) └ [ChTrendlines](#)  
└ [ChTrendline](#)

The collection of [ChTrendline](#) objects for a series.

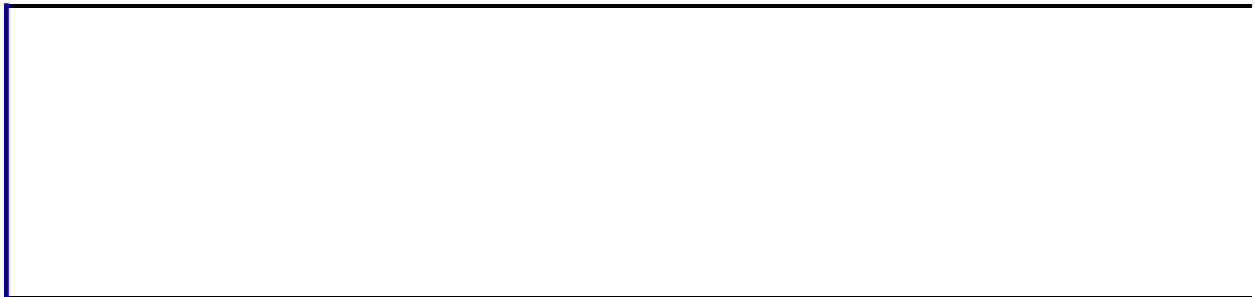
## Using the ChTrendlines collection

The [Trendlines](#) property of the [ChSeries](#) object returns a **ChTrendlines** collection.

Use the **Add** method of the **ChTrendline** object to add a trendline to your chart.

The following example adds a trendline to the first series in the first chart in Chartspace1 and then formats the trendline.

```
Sub AddPolyTrendline()  
  
    Dim serSeries1  
    Dim chConstants  
    Dim tlSeries1Trend  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the first series of the first chart  
    ' in Chartspace1.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a trendline to the first series and return  
    ' a Trendline object.  
    Set tlSeries1Trend = serSeries1.Trendlines.Add  
  
    ' Display the equation used to calculate the trendline.  
    tlSeries1Trend.IsDisplayingEquation = True  
  
    ' Set the trendline to be a polynomial trendline.  
    tlSeries1Trend.Type = chConstants.chTrendlineTypePolynomial  
  
End Sub
```



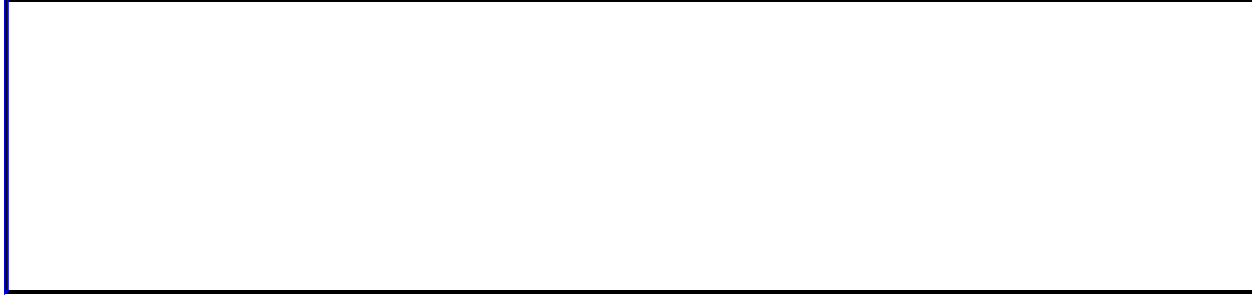
# ChUserDefinedSelection Object

## [ChUserDefinedSelection](#)

Represents an object drawn on the chart between calls to the [BeginObject](#) and [EndObject](#) methods.

## Using the ChUserDefinedSelection object

When a custom drawing object is selected, the [ChartSpace](#) object's [Selection](#) property returns a **ChUserDefinedSelection** object.



# Coordinate Object

Multiple objects  [Coordinate](#)

Stores the X and Y-coordinates of a data point for later retrieval.

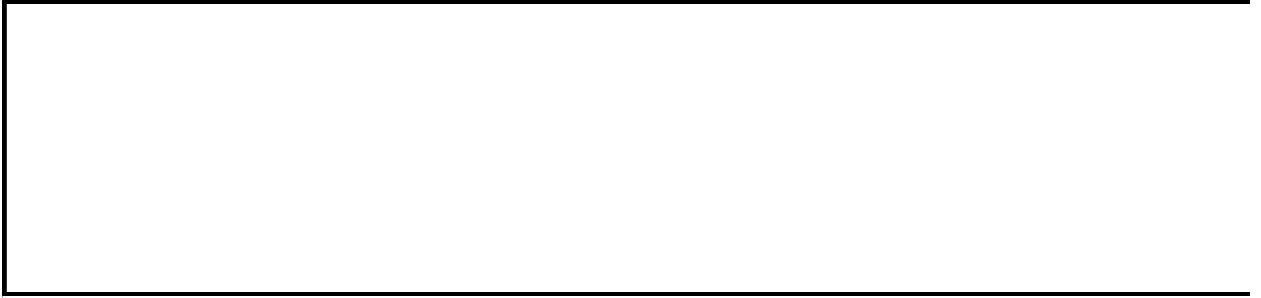
## Using the **Coordinate** object

Use the [ValueToPoint](#) method of the [ChAxis](#) or [ChSeries](#) object to return a **Coordinate** object.

Use the [x](#) and [y](#) properties of the **Coordinate** object to return the X and Y-coordinates of the data point currently stored in the **Coordinate** object.

The following example changes the title of the first chart in Chartspace1 to the pixel coordinates of a data point in the first series of the chart.

```
Sub GetPixelCoordinates()  
  
    Dim chChart1  
    Dim lXPos  
    Dim lYPos  
    Dim coPointCoordinates  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set chChart1 = ChartSpace1.Charts(0)  
  
    ' Enable the title for the chart.  
    chChart1.HasTitle = True  
  
    ' Set a Coordinate object to the coordinates of a data point.  
    Set coPointCoordinates = chChart1.SeriesCollection(0).ValueToPoi  
  
    ' Set a variable to the X-coordinate.  
    lXPos = coPointCoordinates.x  
  
    ' Set a variable to the Y-coordinate.  
    lYPos = coPointCoordinates.y  
  
    ' Set the chart's titles to the pixel coordinates of the specifi  
    ' data point.  
    chChart1.Title.Caption = "X(" & lXPos & ") Y(" & lYPos & ")"  
  
End Sub
```





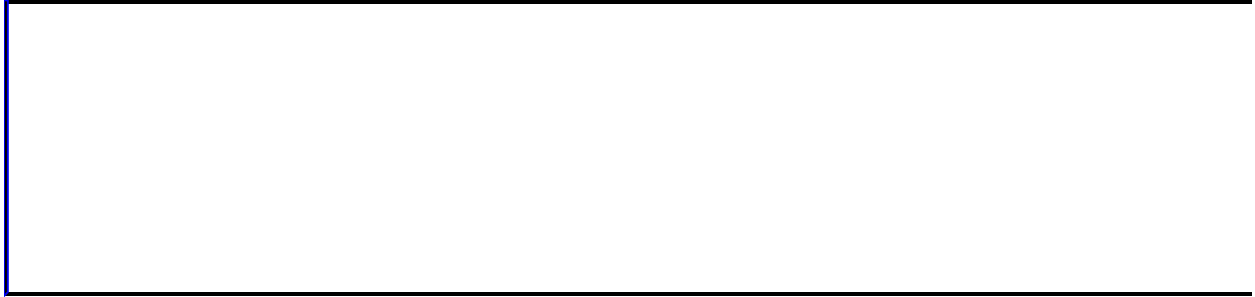
# Criteria Object

[Filter](#) └ [Criteria](#)



Contains the entire array of AutoFilter criteria. Each criterion is a **String** value. The **AutoFilter** object contains a **Filters** collection, the **Filters** collection contains a **Filter** object for each column in the filtered range, and each **Filter** object contains a **Criteria** object.

## Using the Criteria Object

The [Filter](#) object's [Criteria](#) property returns the **Criteria** object for the specified filter.



# DataPage Object

Multiple objects  [DataPage](#)  
 Multiple objects

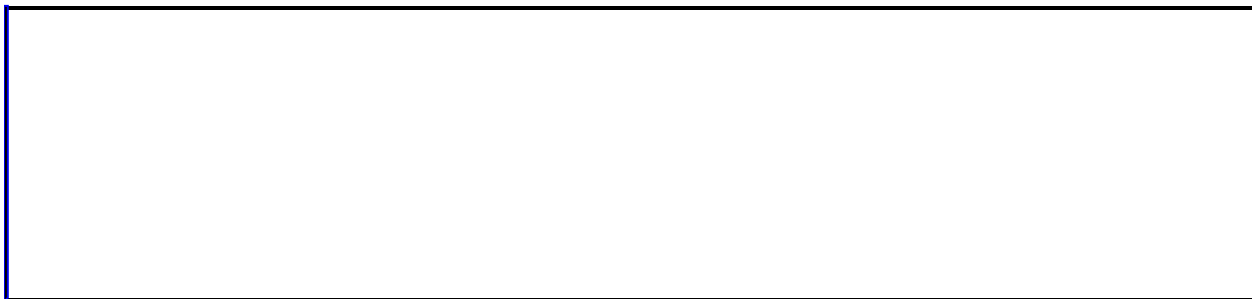
Represents the combination of sections that are shown on a data access page when the expand button is clicked. These sections include a caption section, a group header and group footer section for each record, and a navigation section corresponding to the set of visible records from a recordset within a single parent record.

The **DataPage** object is a member of the [DataPages](#) collection.

## Using the DataPage Object

The following properties return a **DataPage** object:

- The [DataPages](#) object's [Item](#) property
- The [Section](#) object's [DataPage](#) property
- The [DSCEventInfo](#) object's **DataPage** property



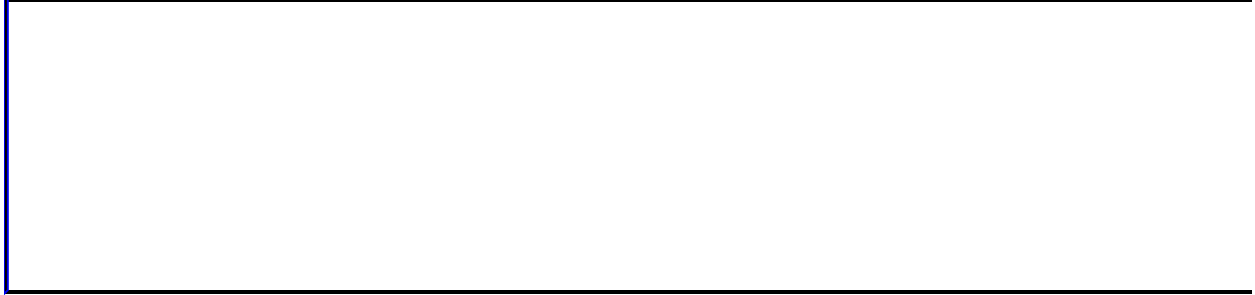
# DataPages Collection Object

[DataSourceControl](#) └ [DataPages](#)  
└ [DataPage](#)

Contains all [DataPage](#) objects in the data source control.

## Using the DataPages Collection

The [DataSourceControl](#) object's [DataPages](#) property returns a **DataPages** collection.



# DataSourceControl Object

[DataSourceControl](#)   └ Multiple objects

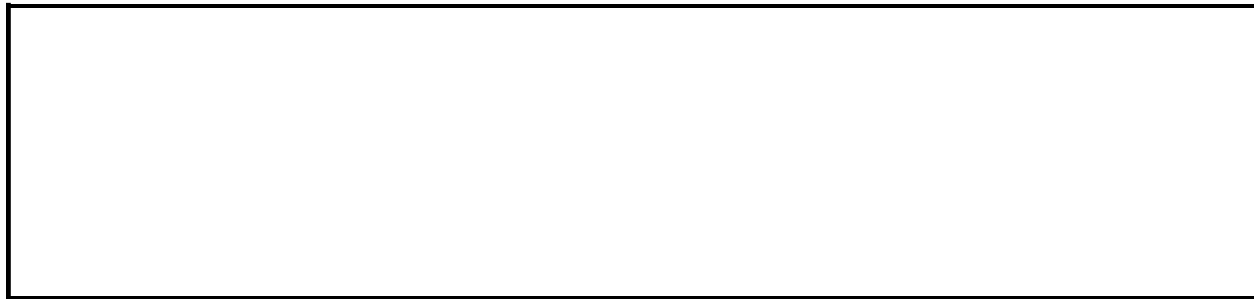
Represents a data source control. The data source control is the top-level container in the data model.

## Using the DataSourceControl Object

You can use either the **CreateObject** method or the **New** keyword to create a **DataSourceControl** object.

The object ID for a data source control on an HTML page returns a **DataSourceControl** object. The programmatic identifier for the **DataSourceControl** object is CLSID:0002E553-0000-0000-C000-000000000046. The following example creates a data source control named "MSODSC" on an HTML page.

```
<object id=MSODSC classid=CLSID:0002E553-0000-0000-C000-000000000046
```





# DSCEventInfo Object

[DSCEventInfo](#)   └ Multiple objects

Contains information about the specified data source control event.

## Using the DSCEventInfo Object

The following data source control events use a **DSCEventInfo** object as their only parameter:

- The [AfterDelete](#) event
- The [AfterInsert](#) event
- The [AfterUpdate](#) event
- The [BeforeCollapse](#) event
- The [BeforeDelete](#) event
- The [BeforeExpand](#) event
- The [BeforeFirstPage](#) event
- The [BeforeInitialBind](#) event
- The [BeforeInsert](#) event
- The [BeforeLastPage](#) event
- The [BeforeNextPage](#) event
- The [BeforeOverwrite](#) event
- The [BeforePreviousPage](#) event
- The [BeforeUpdate](#) event
- The [Current](#) event
- The [DataError](#) event
- The [DataPageComplete](#) event

- The [Dirty](#) event
- The [Focus](#) event
- The [RecordExit](#) event
- The [RecordsetSaveProgress](#) event
- The [Undo](#) event

You can use the properties of the **DSCEventInfo** object to return information about the data access page when an event is trapped. The [Section](#) property can be used to determine the section of the data access page where the event occurred. You can use the [ReturnValue](#) property to cancel the completion of some events.

The events listed above vary in their support of the **DSCEventInfo** properties. Some of the events support a subset of the **DSCEventInfo** properties, and some events don't support any of the **DSCEventInfo** properties. Using an unsupported property will result in a run-time error.

The following example cancels the deletion of a record if the Discontinued field is set the No. The **Section** property of the **DSCEventInfo** object is used to drill down to the value of the Discontinued field. If the field contains the value No, then the **ReturnValue** of the **DSCEventInfo** object is set to **False**, canceling the deletion of the record.

```
Sub MSODSC_BeforeDelete(DSCEventInfo)

    Dim txtDiscontinued

    ' Set a variable to the text box that contains the value
    ' of the Discontinued field for the record that is to be deleted.
    Set txtDiscontinued = DSCEventInfo.Section.HTMLContainer _
        .Children("Discontinued")

    ' Check the value of the control.
    If txtDiscontinued.Value = "No" Then

        ' Display a message to the user.
        MsgBox "Do not delete products that have not " & _
            "been discontinued."
```

```
        Cancel the deletion of the record.  
        DSCEventInfo.ReturnValue = False  
    End If
```

```
End Sub
```



# ElementExtension Object

[ElementExtensions](#) └ [ElementExtension](#)

Adds data-related properties to HTML elements such as text boxes and drop-down list boxes. Element extensions tie recordset and grouping definition objects such as page fields to an element on the HTML page that can be bound to their resulting data.

The **ElementExtension** object is a member of the [ElementExtensions](#) collection.

## Using the ElementExtension Object

The following method and property return an **ElementExtension** object:

- The [ElementExtensions](#) collection's [Add](#) method
- The **ElementExtensions** collection's [Item](#) property



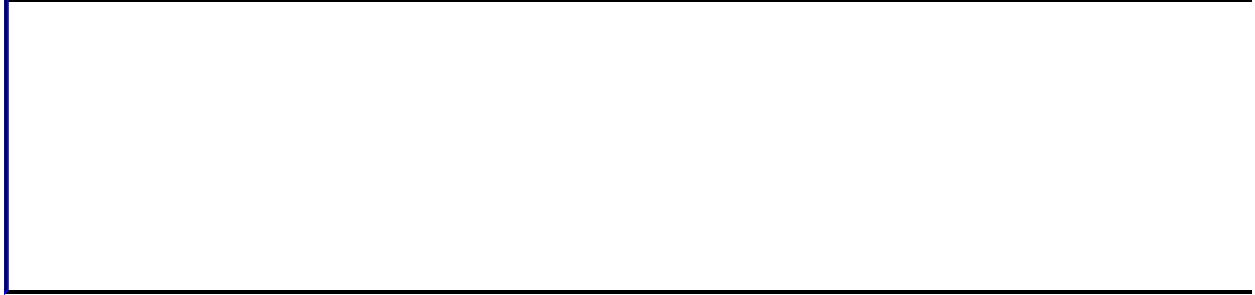
# ElementExtensions Collection Object

[DataSourceControl](#) └ [ElementExtensions](#)  
└ [ElementExtension](#)

Contains the [ElementExtension](#) objects for the specified data source control.

## Using the ElementExtensions Collection

The [DataSourceControl](#) object's [ElementExtensions](#) property returns an **ElementExtensions** collection.





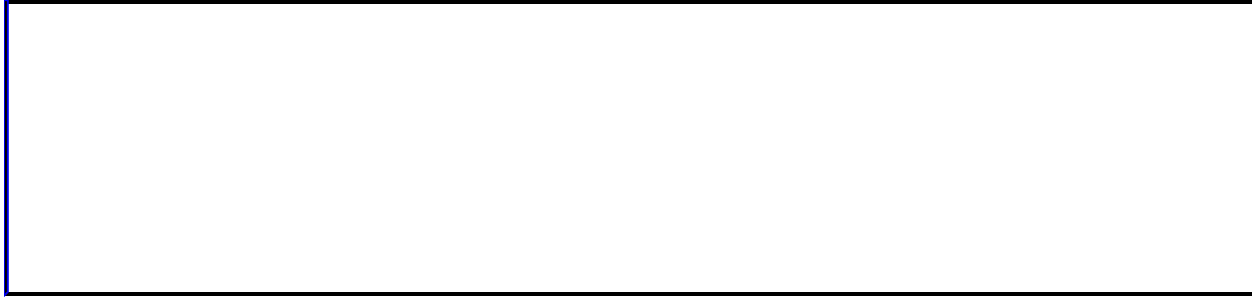
# Filter Object

[Filters](#) └ [Filter](#)  
└ [Criteria](#)

Represents a single filter used with the AutoFilter. The **Filter** object is a member of the **Filters** collection. The **AutoFilter** object contains a **Filters** collection, the **Filters** collection contains a **Filter** object for each column in the filtered range, and each **Filter** object contains a **Criteria** object.

## Using the Filter Object

The [Filters](#) object's [Item](#) property returns a **Filter** object.



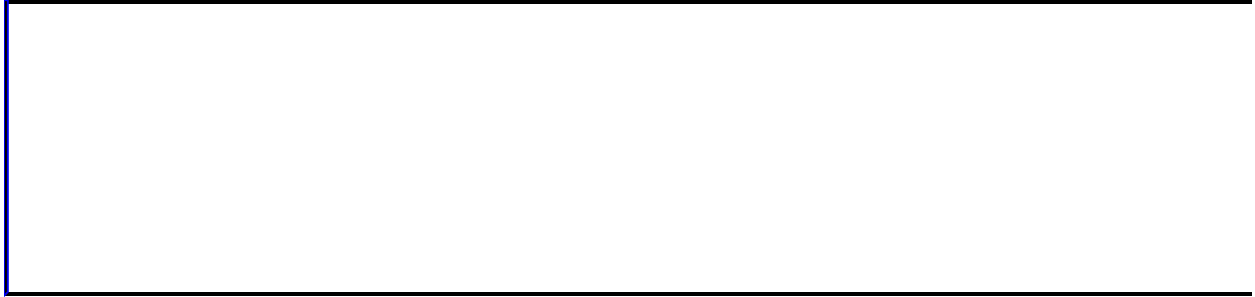
# Filters Collection Object

[AutoFilter](#) └ [Filters](#)  
└ [Filter](#)

Represents the collection of filters used with the AutoFilter. Each filter is represented by a [Filter](#) object. The **AutoFilter** object contains a **Filters** collection, the **Filters** collection contains a **Filter** object for each column in the filtered range, and each **Filter** object contains a **Criteria** object.

## Using the Filters Object

The [AutoFilter](#) object's [Filters](#) property returns a **Filters** collection.



# Font Object

Multiple objects  [Font](#)

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 a **Font** object.

The following example formats cells A1:C5 as bold.

```
Spreadsheet1.ActiveSheet.Range("A1:A5").Font.Bold = True
```



# GroupingDef Object

[AllGroupingDefs](#) └ [GroupingDef](#)

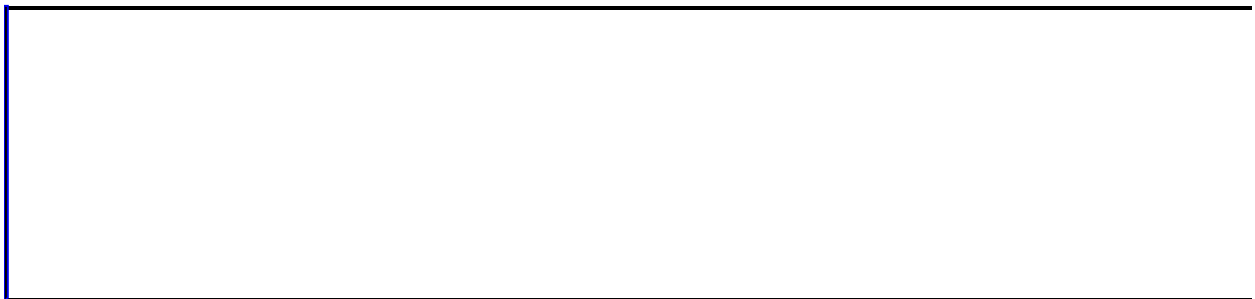
Represents a grouping definition. A grouping definition is a special type of recordset definition. As with [RecordsetDef](#) objects, you can use a grouping definition's name in the [Execute](#) method and in the [RecordSource](#) property of a group level or element extension.

A grouping definition defines an ADO grouping recordset that appears as a parent of the containing recordset in the hierarchy or recordsets produced by a page. At run time, all the data for the containing recordset definition is fetched before the grouping recordset is created.

## Using the GroupingDef Object

The following methods and property return a **GroupingDef** object:

- The [GroupingDefs](#) collection's [Add](#) method
- The **GroupingDefs** collection's [AddTotal](#) method
- The **GroupingDefs** or [AllGroupingDefs](#) collections' [Item](#) property





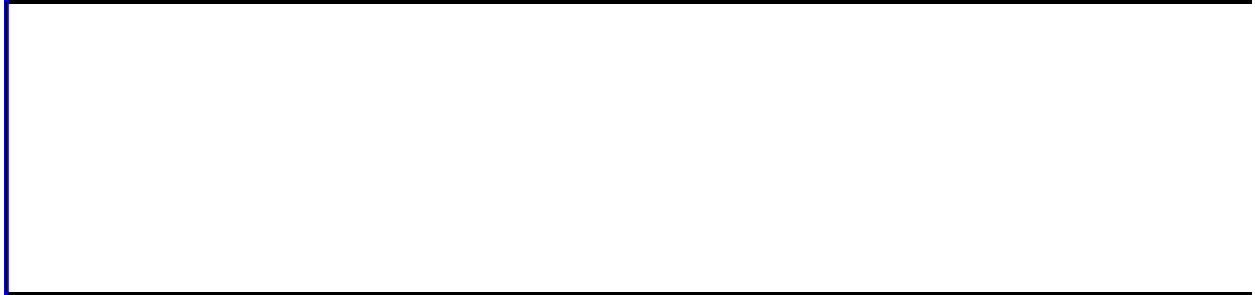
# GroupingDefs Collection Object

[RecordsetDef](#) └ [GroupingDefs](#)  
└ [GroupingDef](#)

Represents the collection of [GroupingDef](#) objects that create grouping parents of the detail records in a recordset definition. Sequence is important: the lower the index, the higher the level of grouping. Index 0 (zero) is the lowest grouping level.

## Using the GroupingDefs Collection

The [RecordsetDef](#) object's [GroupingDefs](#) property returns a **GroupingDefs** collection.



# GroupLevel Object

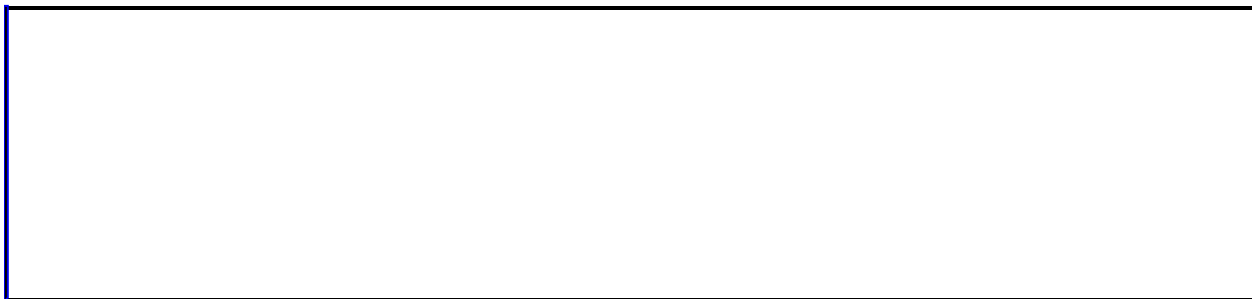
Multiple objects [└ GroupLevel](#)

Represents the set of all records at a given level of the data access page hierarchy. The **GroupLevel** object is a member of the [GroupLevels](#) collection.

## Using the GroupLevel Object

The following method and property return a **GroupLevel** object:

- The [GroupLevels](#) collection's [Add](#) method
- The **GroupLevels** collection's [Item](#) property
- The [DataPage](#) object's [GroupLevel](#) property



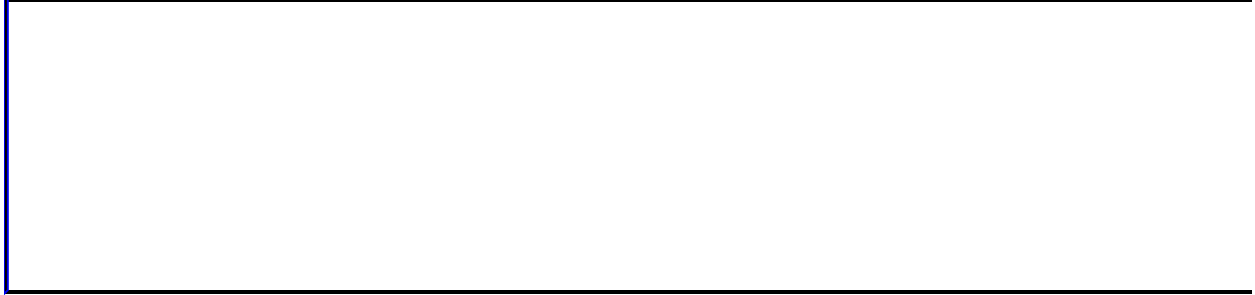
# GroupLevels Collection Object

[DataSourceControl](#) └ [GroupLevels](#)  
└ [GroupLevel](#)

Represents the collection of [GroupLevel](#) objects for the specified data source control. Each **GroupLevel** object represents the set of all records at a given level of the data access page hierarchy.

## Using the GroupLevels Collection

The [DataSourceControl](#) object's [GroupLevels](#) property returns a **GroupLevels** collection.



# Heading Object

[Headings](#) └ [Heading](#)

Represents a single row or column header in the specified window's [Headings](#) collection.

## Using the Heading object

Use the **Headings** collection's **Item** property to return an individual **Heading** object. This can be expressed as `Headings(Index)` or `Headings.Item(Index)`, where *Index* is the index number of the individual **Heading** object.

Use the [Caption](#) property to customize the row and column headings in a window. The following example illustrates how to use the **Caption** property to change the row and column heading in a window:

```
Sub Change_Headings()  
  
    Dim hdrColHeadings  
    Dim hdrRowHeadings  
  
    ' Set a variable to the column headings in the active window.  
    Set hdrColHeadings = Spreadsheet1.ActiveWindow.ColumnHeadings  
  
    ' Set a variable to the row headings in the active window.  
    Set hdrRowHeadings = Spreadsheet1.ActiveWindow.RowHeadings  
  
    ' Set the headings of columns A through D.  
    hdrColHeadings(1).Caption = "Qtr 1"  
    hdrColHeadings(2).Caption = "Qtr 2"  
    hdrColHeadings(3).Caption = "Qtr 3"  
    hdrColHeadings(4).Caption = "Qtr 4"  
  
    ' Set the headings of rows 1 through 5.  
    hdrRowHeadings(1).Caption = "Sedan"  
    hdrRowHeadings(2).Caption = "Convertible"  
    hdrRowHeadings(3).Caption = "Truck"  
    hdrRowHeadings(4).Caption = "Sport-Utility"  
    hdrRowHeadings(5).Caption = "Minivan"  
  
End Sub
```

The **Caption** property of the **Heading** object is limited to 256 characters, and the text cannot be wrapped to a second line.

Use the [ResetHeadings](#) method to set the row and column headings back to their default values.





# Headings Object

[Window](#) └ [Headings](#)  
└ [Heading](#)

A collection of the row and column headings for a [Window](#) object.

## Using the Headings object

The following properties return a **Headings** collection.

The [Window](#) object's [RowHeadings](#) property.


The **Window** object's [ColumnHeadings](#) property.

The following example customizes the heading of column D in the active sheet in Spreadsheet1:

```
Spreadsheet1.ActiveWindow.ColumnHeadings(4).Caption = "1999 Sales"
```



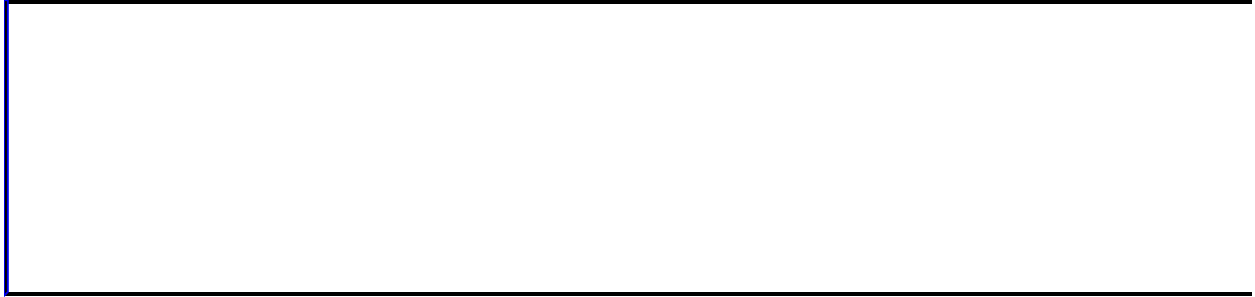
# Hyperlink Object

[Range](#)  [Hyperlink](#)

Represents a hyperlink.

## Using the Hyperlink Object

The [Range](#) object's [Hyperlink](#) property returns a **Hyperlink** object.



# Interior Object

Multiple objects  [Interior](#)

Represents the interior of an object.

## Using the Interior Object

The following properties return an **Interior** object:

- The [Range](#) object's **Interior** property
- The [TitleBar](#) object's **Interior** property



# LookupRelationships Collection Object

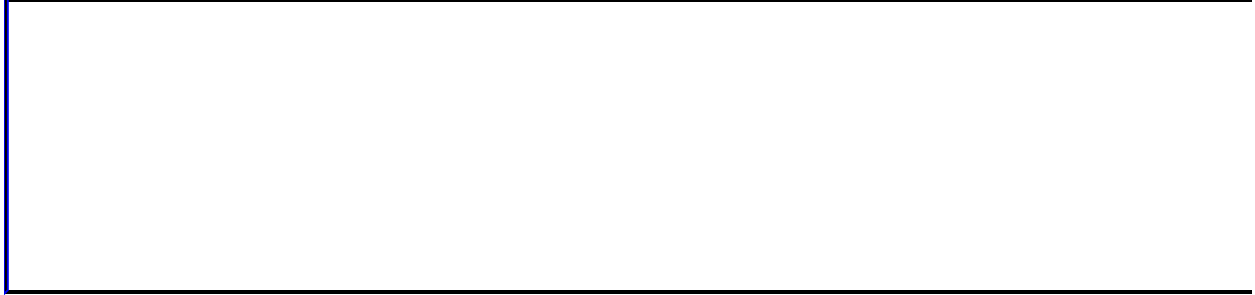
[PageRowsource](#) └ [LookupRelationships](#)  
└ [PageRelationship](#)

Represents the collection of [PageRelationship](#) objects that define a lookup join relationship with a page row source. In the object diagram shown in this topic, the first **PageRowsource** object (above the **LookupRelationships** collection) is the object on the “many” side of the one-to-many relationship and the **PageRelationship** object (below the collection) is the object on the “one” side.



## Using the LookupRelationships Collection

The [PageRowsource](#) object's [LookupRelationships](#) property returns a **LookupRelationships** collection.



# LookupSchemaRelationships Collection Object

[SchemaRowsource](#) └ [LookupSchemaRelationships](#)  
└ [SchemaRelationship](#)



Represents the collection of [SchemaRelationship](#) objects for which a single schema row source acts as the “many” side of the one-to-many relationship.

## Using the LookupSchemaRelationships Collection

The [SchemaRowsource](#) object's [LookupSchemaRelationships](#) property returns a **LookupSchemaRelationships** collection.



# Name Object

Multiple objects  [Name](#)  
 [Range](#)

Represents a defined name for a range of cells, a formula, or a constant value.  
The **Name** object is a member of the [Names](#) collection.

## Using the Name object

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. 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
```

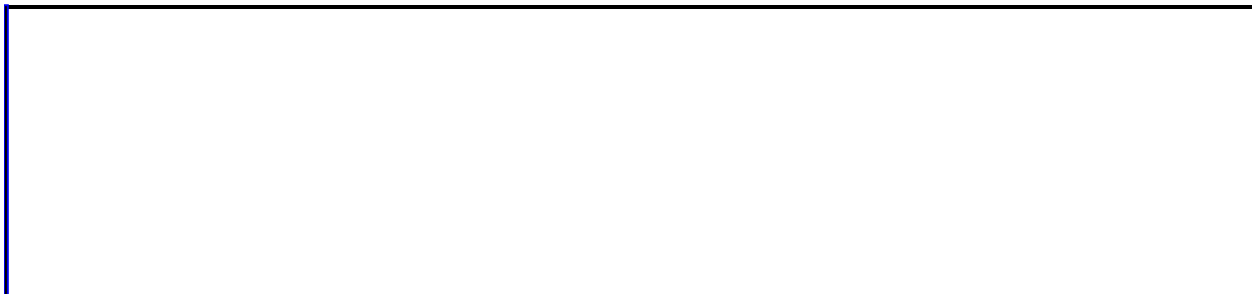
Use the **Name** property to return or set the text of the name itself. The following example changes the name of the first **Name** object in the active workbook.

```
Names(1).Name = "stock_values"
```

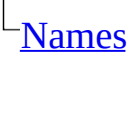

Use the [Add](#) method to create a name and add it to the collection. The following example creates a new name that refers to cells A1:C20 on the worksheet named "Sheet1."

```
Spreadsheet1.Names.Add "CurrentMonth", "=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".



# Names Collection

Multiple objects  [Names](#)  
 [Name](#)

A collection of all the [Name](#) objects in the workbook. Each **Name** object can represent a defined name for a range of cells, a formula, or a constants value.

## Using the Names collection

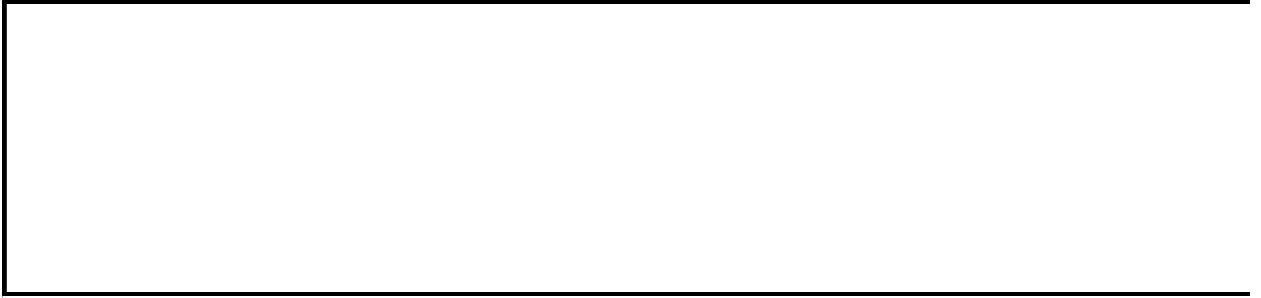
Use the **Names** property to return the **Names** collection. The following example creates a list of all the names in the active workbook, along with the addresses to which they refer.

```
Sub List_All_Names()  
    Dim nmCurrentName  
    Dim rngCurrent  
  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Range("A1")  
  
    ' Loop through all of the names in the active workbook.  
    For Each nmCurrentName In Spreadsheet1.ActiveWorkbook.Names  
  
        ' Write the current name to the worksheet.  
        rngCurrent.Value = nmCurrentName.Name  
  
        ' Write the definition of the current name to the worksheet.  
        rngCurrent.Offset(0, 1).Value = "" & nmCurrentName.RefersTo  
  
        Set rngCurrent = rngCurrent.Offset(1, 0)  
    Next  
End Sub
```

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."


```
Spreadsheet1.Names.Add "CurrentMonth", "=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".





# OCCommand Object

[OCCommands](#)  [OCCommand](#)

Represents a single command in the specified Microsoft Office Web Component.

## Using the **OCCommand** object

Use the [Item](#) property of the **OCCommands** collection to return a single **OCCommand** object.

The **OCCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each Microsoft Office Web Component.

Use the [Execute](#) method of the **OCCommand** object to execute a particular command. The following example uses the **Execute** method to select the upper-left cell in the active sheet of Spreadsheet1.

```
Sub SelectUpperLeft()  
    Dim ssConstants  
    Set ssConstants = Spreadsheet1.Constants  
    ' Select the upper-left cell in the active worksheet.  
    Spreadsheet1.Commands(ssConstants.ssCommandMoveToOrigin).Execute  
End Sub
```

Use the [CommandBeforeExecute](#) event to impose certain restrictions before a command is executed, or to cancel a command. The following example refreshes PivotTable1 when the Export command is invoked so that the latest data is exported to Microsoft Excel.

```
Sub PivotTable1_CommandBeforeExecute(Command, Cancel)  
    Dim ptConstants  
    Set ptConstants = PivotTable1.Constants  
    ' Check to see if the Export command  
    ' has been invoked.  
    If Command = ptConstants.plCommandExport Then  
        ' Refresh the PivotTable list.  
        PivotTable1.Refresh  
    End If
```

End Sub

The following example prevents the user from cutting, copying, or exporting the contents of Spreadsheet1 to Microsoft Excel.

```
Sub Spreadsheet1_CommandBeforeExecute(Command, Cancel)

    Dim ssConstants

    Set ssConstants = Spreadsheet1.Constants

    Select Case Command

        ' Check to see if the Export command has
        ' been invoked.
        Case ssConstants.ssCommandExport

            ' Cancel the command.
            Cancel.Value = True

            ' Display a message to the user.
            MsgBox "Export of the data is prohibited."

        ' Check to see if the Cut or Copy commands
        ' have been invoked.
        Case ssConstants.ssCommandCopy, ssConstants.ssCommandCut

            ' Cancel the command.
            Cancel.Value = True

            ' Display a message to the user.
            MsgBox "Cutting or Copying the data is prohibited."
    End Select

End Sub
```

Use the [CommandExecute](#) event when you want to execute a set of commands when a particular command is executed. The following example writes the current date and time to an HTML text box control every time that PivotTable1 is refreshed.

```
Sub PivotTable1_CommandExecute(Command)

    Dim ptConstants

    Set ptConstants = PivotTable1.Constants

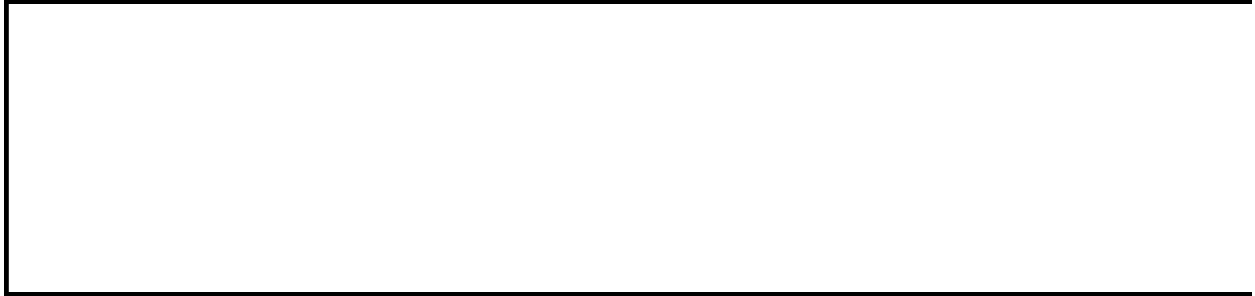
    ' Check to see if the PivotTable list has been refreshed.
```

```
If Command = ptConstants.plCommandRefresh Then
```



```
    ' Write the current data and time to the text box.  
    TextBox.Value = "PivotTable last refreshed on " & _  
                   Date & " at " & Time
```

```
End If
```

```
End Sub
```



# OCCommands Object

Multiple objects  [OCCommands](#)  
 [OCCommand](#)

Contains a collection of [OCCommand](#) objects that represent the collection of all of the user interface and keyboard commands that are available in the specified Microsoft Office Web Component.

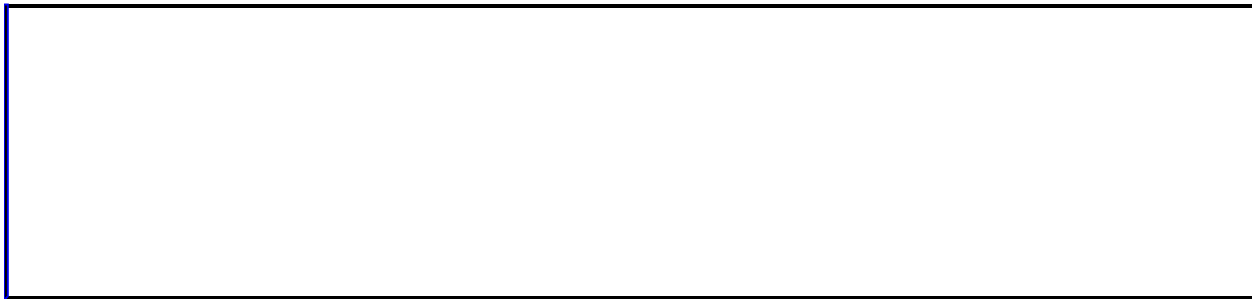
## Using the OCommands object

The following properties return a **OCommands** collection:

- The [ChartSpace](#) object's [Commands](#) property.
- The [PivotTable](#) object's **Commands** property.
- The [Spreadsheet](#) object's **Commands** property.

The **OCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each Web component.

Use the [Item](#) property to return a single **OCommand** object.



# OWCLanguageSettings Object

Multiple objects [OWCLanguageSettings](#)

Returns information about the language settings for the Microsoft Office Web Components.

## Using the OWCLanguageSettings object

Use **Application.LanguageSettings.LanguageID(*MsoAppLanguageID*)**, where *MsoAppLanguageID* is one of the following constants used to return locale identifier (LCID) information to the specified application:

**msoLanguageIDHelp**, **msoLanguageIDInstall**, **msoLanguageIDUI**, or **msoLanguageIDUIPrevious**. The following example returns the install language, user interface language, and Help language LCIDs for a spreadsheet control named Spreadsheet1 in a message box.

```
MsgBox "The following locale IDs are registered " & _  
    "for this application: Install Language - " & _  
    Spreadsheet1.LanguageSettings.LanguageID(msoLanguageIDInstall) &  
    " User Interface Language - " & _  
    Spreadsheet1.LanguageSettings.LanguageID(msoLanguageIDUI) & _  
    " Help Language - " & _  
    Spreadsheet1.LanguageSettings.LanguageID(msoLanguageIDHelp)
```



The following example tests whether the U.S. English language is registered as a preferred editing language.

```
If Spreadsheet1.LanguageSettings. _  
    LanguagePreferredForEditing(msoLanguageIDEnglishUS) Then  
    MsgBox "U.S. English is one of the chosen editing languages."  
End If
```





# PageField Object

Multiple objects  [PageField](#)  
 Multiple objects

Represents a field in a recordset or grouping definition. The page field name is used by a control on a data access page to bind to data from a recordset.

**PageField** objects come in three types:

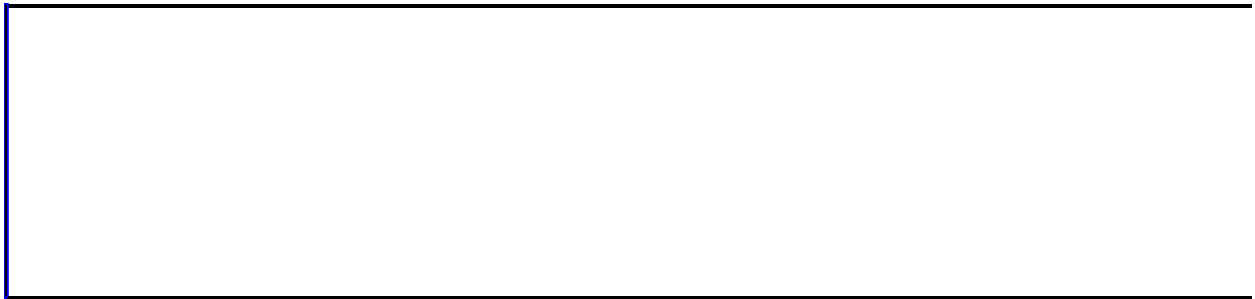
- **Output.** This type comes directly from a schema field in a schema row source. For schema row sources of the table type or view type, this means the field is listed in the SELECT list of the SQL statement generated by the data model. For schema row sources of the text type or stored procedure type, all schema fields appear as output and the page field name must be the same as the schema field name.
- **Calculated.** This is a locally calculated column added to a recordset. Visual Basic for Applications expression syntax is supported. Expressions can reference page fields of type **dscOutput** or **dscGrouping** within the same recordset definition or grouping definition. Expressions can also reference HTML elements by using the document object model (for example, `=Quantity*UnitPrice*Document.All("Text0").Value`). Calculated fields are recalculated whenever an updated record is saved, whenever the page is refreshed, or whenever the recordset's **Resync** method is called.
- **Grouping.** This is a grouping field or aggregate field attached to a **GroupingDef** or **RecordsetDef** object.

The **PageField** object can be a member of the **AllPageFields**, **GroupingFields**, **OutputFields**, or **PageFields** collection.



## Using the PageField Object

The following properties and method return a **PageField** object:

- The [AllPageFields](#) collection's [Item](#) property
- The [PageFields](#) collection's [Add](#) method
- The **PageFields** collection's **Item** property
- The [PageRelatedField](#) object's [ManySide](#) property
- The **PageRelatedField** object's [OneSide](#) property



# PageFields Collection Object

Multiple objects  [PageFields](#)  
 [PageField](#)

The collection of **PageField** objects describing the fields that will be present in the recordset produced by the specified recordset definition. These fields are a combination of output fields from the page row sources that make up this recordset definition and other page fields that are added to this collection. Sequence within this collection is unimportant because all controls bind by name rather than by their ordinal relationship to fields in the output recordset.

## Using the PageFields Collection Object

The [RecordsetDef](#), [PageRowSource](#), and [GroupingDef](#) objects' [PageFields](#) property returns a **PageFields** collection.



# PageRelatedField Object

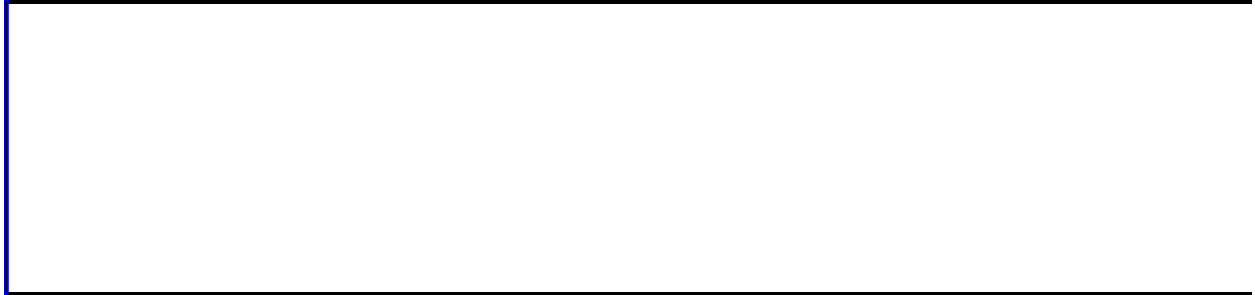
[PageRelatedFields](#) └ [PageRelatedField](#)  
└ [PageField](#)

Represents a page instance of the column pairings that make up a page relationship. The names of these fields are used in generating join clauses in SQL and in relating a parent recordset definition to a child recordset definition.

The [PageRelatedField](#) object is a member of the [PageRelatedFields](#) collection.

## Using the PageRelatedField Object

The [PageRelatedFields](#) collection's [Item](#) property returns a **PageRelatedField** object.



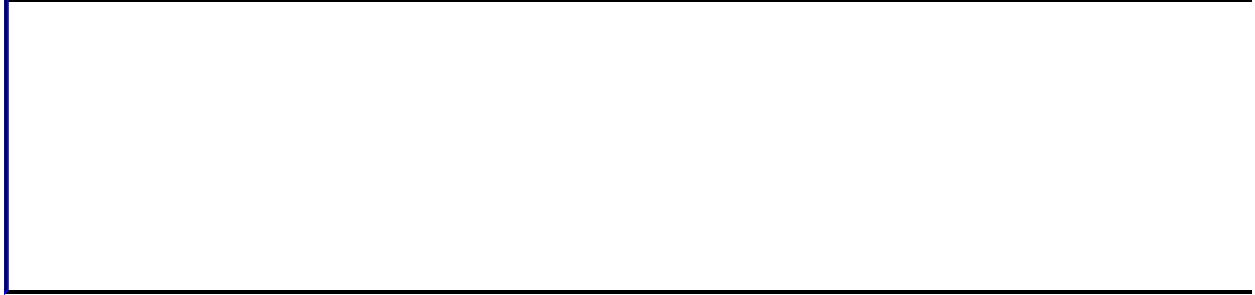
# PageRelatedFields Collection Object

[PageRelationship](#) └ [PageRelatedFields](#)  
└ [PageRelatedField](#)

The collection of **[PageRelatedField](#)** objects that belong to a given recordset definition. This collection is automatically filled when a recordset definition based on the **[SchemaRelatedFields](#)** collection is created.



## Using the PageRelatedFields Collection

The [PageRelationship](#) object's [PageRelatedFields](#) property returns a **PageRelatedFields** collection.





# PageRelationship Object

Multiple objects  [PageRelationship](#)  
 Multiple objects

A **PageRelationship** object ties two page row sources together, either within a recordset definition (a **LookupRelationships** collection) or between recordset definitions (a **SublistRelationships** collection). A page relationship is created from information in a [SchemaRelationship](#) object.

The **PageRelationship** object is a member of the **LookupRelationships** or **SublistRelationships** collection.



## Using the PageRelationship Object

The following methods and properties return a **PageRelationship** object:

- The [LookupRelationships](#) collection's [Add](#) method
- The **LookupRelationships** collection's [Item](#) property
- The [SublistRelationships](#) collection's **Add** method
- The **SublistRelationships** collection's **Item** property



# PageRowsource Object

Multiple objects  [PageRowsource](#)  
 Multiple objects

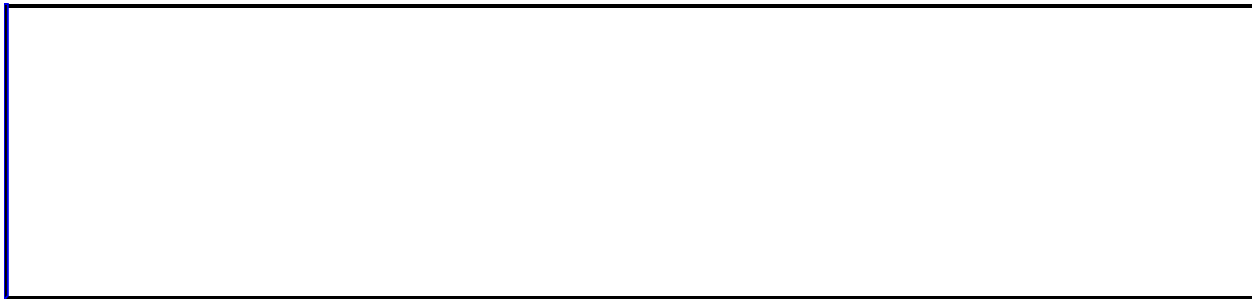
A **PageRowsource** object refers to an instance of a [SchemaRowsource](#) object that is currently in use on a data access page. A table, view, or stored procedure must be in the data model as a schema row source before it can be added as a page row source; the page row source is then used as a data source for the page.

The **PageRowsource** object is a member of the [PageRowsources](#) collection.



## Using the PageRowsource Object

The following properties return a **PageRowsource** object:

- The [PageField](#) object's [PageRowsource](#) property
- The [PageRelationship](#) object's [ManySide](#) property
- The **PageRelationship** object's [OneSide](#) property
- The [PageRowsources](#) collection's [Item](#) property
- The [RecordsetDef](#) object's [PrimaryPageRowsource](#) property



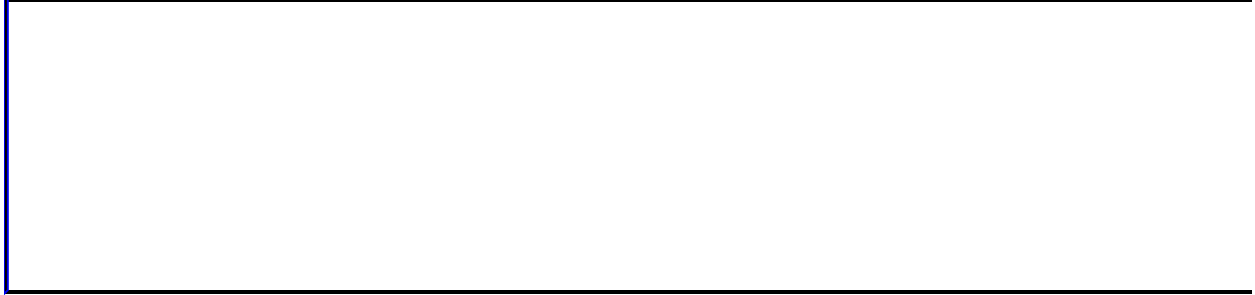
# PageRowsources Collection Object

Multiple objects  [PageRowsources](#)  
 [PageRowsource](#)

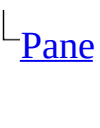
The collection of [PageRowsource](#) objects in the data model.

## Using the PageRowsources Collection Object

The [RecordsetDef](#) object's [PageRowsources](#) property returns a **PageRowsources** collection.



# Pane Object

Multiple objects  [Pane](#)  
[Range](#)

Represents a pane in a window. The **Pane** object is a member of both the [Panes](#) collection and the [Window](#) object.

## Using the Pane Object

The following properties return a **Pane** object:

- The **Panes** collection's [Item](#) property
- The **Window** object's [ActivePane](#) property





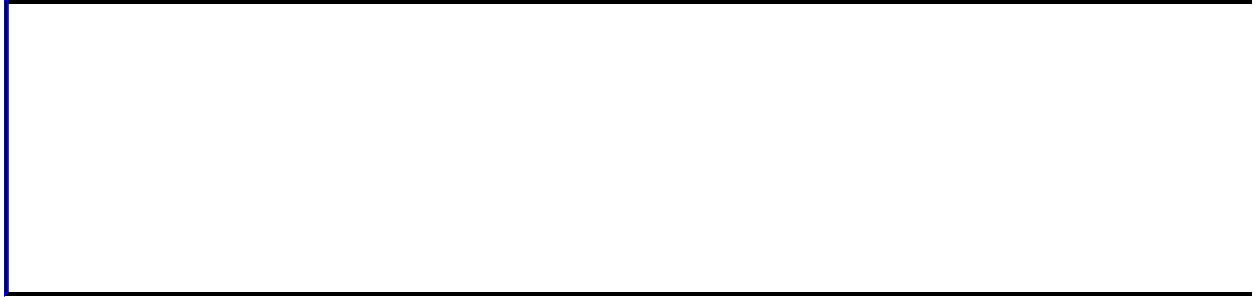
# Panes Collection Object

[Window](#) └ [Panels](#)  
└ [Pane](#)

The collection of [Pane](#) objects for a worksheet.

## Using the Panes Collection

The [Window](#) object's [Panels](#) property returns a **Panels** collection.



# ParameterValue Object

[ParameterValues](#) └ [ParameterValue](#)

Represents an input parameter value expression. This expression is evaluated at execute time to provide a run-time parameter value to a row source of type **dscProcedure** or **dscCommandText**.

## Using the ParameterValue Object

The following method and property return a **ParameterValue** object:

- The [ParameterValues](#) collection's [Add](#) method
- The **ParameterValues** collection's [Item](#) property



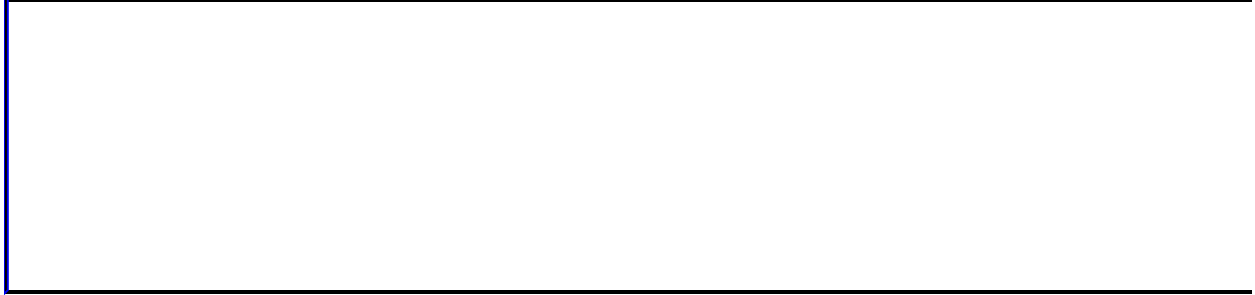
# ParameterValues Collection Object

[RecordsetDef](#) └ [ParameterValues](#)  
└ [ParameterValue](#)

The collection of [\*\*ParameterValue\*\*](#) objects for a recordset definition.

## Using the ParameterValues Collection

The [RecordsetDef](#) object's [ParameterValues](#) property returns a **ParameterValues** collection.



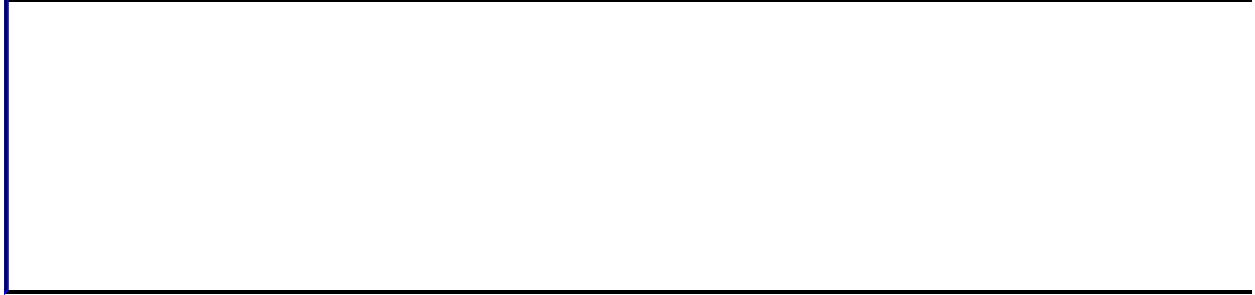
# PivotAggregate Object

[PivotAggregates](#) └ [PivotAggregate](#)  
└ Multiple objects

Represents the data associated with a total in a PivotTable list. A total defines what the user wants to see, but the data that results from the total is called the *aggregate* or *aggregate value*. The **PivotAggregate** object is a member of the **PivotAggregates** collection.

## Using the PivotAggregate Object

The [PivotAggregates](#) collection's [Item](#) property returns a **PivotAggregate** object.





# PivotAggregates Collection Object

[PivotCell](#) └ [PivotAggregates](#)  
└ [PivotAggregate](#)



The collection of [PivotAggregate](#) objects for the specified cell.

## Using the PivotAggregates Collection

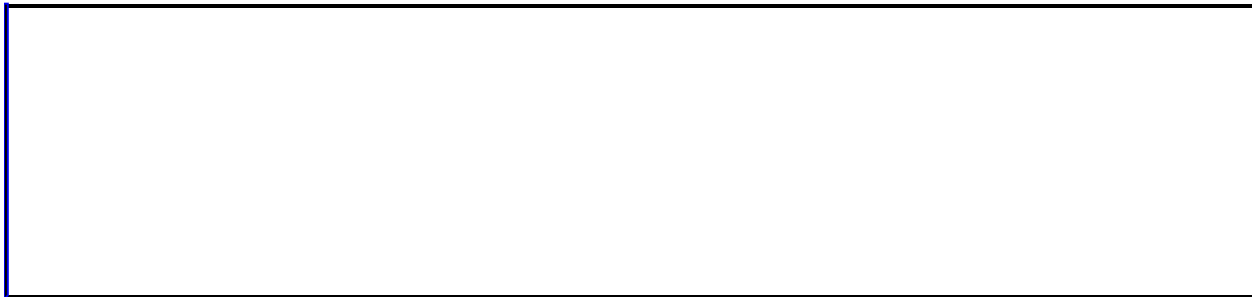
The [PivotCell](#) object's [Aggregates](#) property returns an object from the **PivotAggregates** collection.





# PivotAxis Object

Multiple objects  [PivotAxis](#)  
 Multiple objects

Used as the base class for the [PivotResultAxis](#), [PivotResultColumnAxis](#), [PivotResultDataAxis](#), [PivotResultFilterAxis](#), [PivotResultGroupAxis](#), [PivotResultPageAxis](#), and [PivotResultRowAxis](#) objects. Use the [SourceAxis](#) property of one of these objects to return a **PivotAxis** object.



# PivotAxisMember Object

Multiple objects  [PivotAxisMember](#)  
 Multiple objects

Represents the values displayed for a grouped field. The **PivotAxisMember** object is a member of the [PivotAxisMembers](#) collection.

## Using the PivotAxisMember object

The following properties return a **PivotAxisMember** object:

The [FindAxisMember](#) property of the **PivotAxisMember**, **PivotColumnMember**, **PivotPageMember**, and **PivotRowMember** objects

The [ParentAxisMember](#) property of the **PivotAxisMember**, **PivotColumnMember**, **PivotPageMember**, and **PivotRowMember** objects

The [TotalMember](#) property of the **PivotAxisMember**, **PivotColumnMember**, **PivotPageMember**, and **PivotRowMember** objects



The [Member](#) property of the **PivotResultColumnAxis**, **PivotResultGroupAxis**, **PivotResultPageAxis**, and **PivotResultRowAxis** objects

The [Item](#) property of the **PivotAxisMembers** collection.

The **PivotAxisMember** has many properties in common with the [PivotMember](#) object. However the **PivotAxisMember** object contains some properties that the **PivotMember** object does not have. You can use the [GroupField](#) and [CustomGroupType](#) properties to access the grouping settings of the member. You can use the [Hyperlink](#) property to access the hyperlink settings of the member. The [MemberProperties](#) property can be used to access any member properties of the member. The [Height](#), [Left](#), and [Width](#) properties can be used to set the size and position of the member.



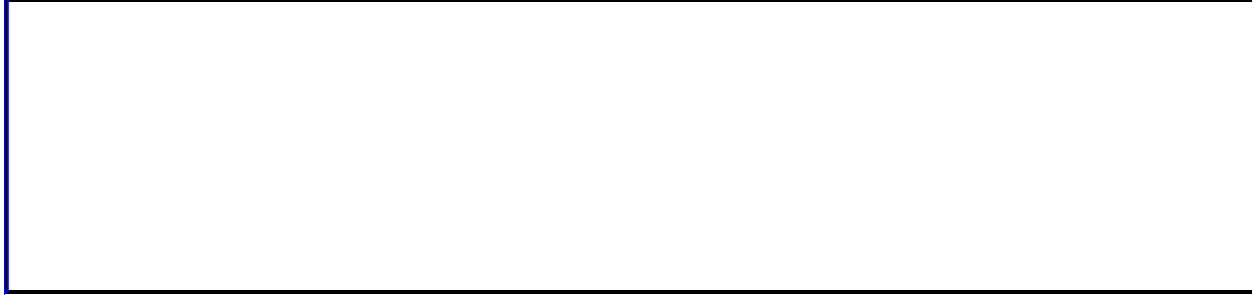
# PivotAxisMembers Collection

Multiple objects  [PivotAxisMembers](#)  
 [PivotAxisMember](#)



A collection of [PivotAxisMember](#) objects.

## Using the PivotAxisMembers collection

Use the [Item](#) property of the **PivotAxisMembers** collection to return a **PivotAxisMember** object.



# PivotCell Object

Multiple objects  [PivotCell](#)  
 Multiple objects

Represents a cell (a grouping of data) in a PivotTable list. A cell displays aggregates, and if the underlying detail records are available, you can have the cell display a detail grid. Grouped fields on the row and column axis determine the amount of data that a given cell represents. Even a simple list with no grouped fields is really a single cell displaying a detail grid.





## Using the PivotCell Object

The following properties return a **PivotCell** object:

- The [PivotAggregate](#), [PivotDetailCell](#), and [PivotDetailRange](#) objects' [Cell](#) property
- The [PivotData](#) object's [Cells](#), [CellsEx](#), and [CurrentCell](#) properties
- The [PivotRange](#) object's [BottomRight](#), [Cells](#), and [TopLeft](#) properties



# PivotColumnMember Object

Multiple objects  [PivotColumnMember](#)  
 Multiple objects

Represents the values displayed for a grouped field in the column area of a PivotTable list. The **PivotColumnMember** object is a member of the [PivotColumnMembers](#) collection.

## Using the PivotColumnMember object

The following properties return a **PivotColumnMember** object:

The [PivotCell](#) object's [ColumnMember](#) property

The [PivotResultColumnAxis](#) object's **ColumnMember** property

The **PivotColumnMember** object's [FindColumnMember](#) property

The **PivotColumnMember** object's [ParentColumnMember](#) property

The **PivotColumnMember** object's [TotalColumnMember](#) property



The **PivotColumnMembers** collection's [Item](#) property

The [PivotData](#) object's [Left](#) property

The **PivotColumnMember** object has many properties in common with the [PivotAxisMember](#) object. Use the [DetailLeft](#), [DetailLeftOffset](#), [DetailsExpanded](#) properties or [MoveDetailLeft](#) method to customize the way detail records are displayed.



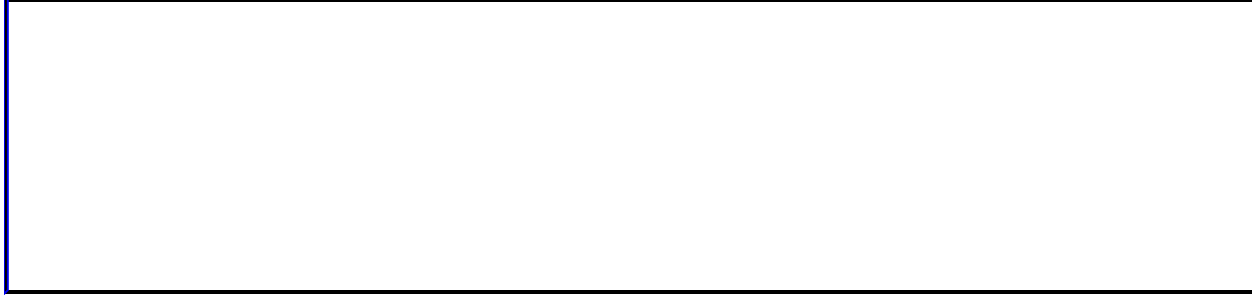
# PivotColumnMembers Collection

Multiple objects  [PivotColumnMembers](#)  
 [PivotColumnMember](#)



A collection of [PivotColumnMember](#) objects.

## Using the PivotColumnMembers Collection

Use the [Item](#) property of the **PivotColumnMembers** collection to return a **PivotColumnMember** object.



# PivotData Object

Multiple objects  [PivotData](#)  
 Multiple objects

Represents the data in a PivotTable list.



## Using the PivotData Object

The following properties return a **PivotData** object:

- The [Data](#) property of the following objects: [PivotCell](#), [PivotResultAxis](#), [PivotResultColumnAxis](#), [PivotResultDataAxis](#), [PivotResultFilterAxis](#), [PivotResultGroupAxis](#), [PivotResultPageAxis](#), and [PivotResultRowAxis](#).
- The [PivotTable](#) object's [ActiveData](#) property.



# PivotDataAxis Object

Multiple objects  [PivotDataAxis](#)  
 Multiple objects

Represents the data axis for a PivotTable list. The data axis contains field sets and totals.





## Using the PivotDataAxis Object

The following properties return a **PivotDataAxis** object:

- The [PivotView](#) object's [DataAxis](#) property
- The [PivotResultDataAxis](#) object's [SourceDataAxis](#) property



# PivotDetailCell Object

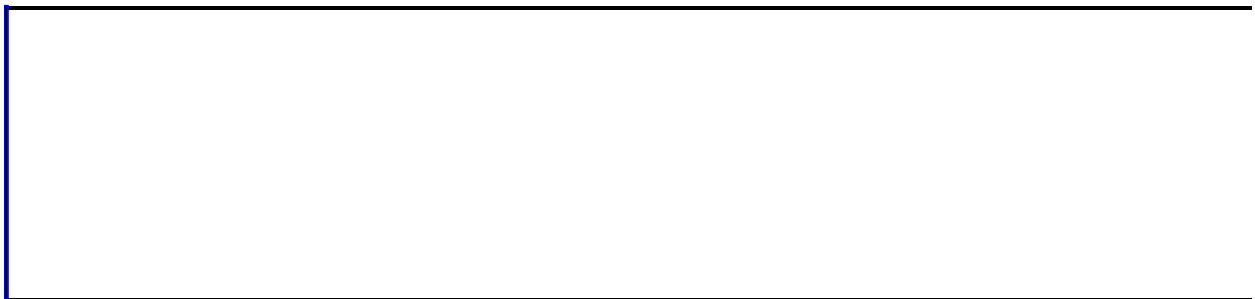
Multiple objects  [PivotDetailCell](#)  
 Multiple objects

Represents a cell in the detail grid for a PivotTable list.

## Using the PivotDetailCell Object

The following properties return a **PivotDetailCell** object:

- The [PivotCell](#) object's [DetailCells](#) property
- The [PivotDetailRange](#) object's [BottomRight](#) property
- The **PivotDetailRange** object's [TopLeft](#) property



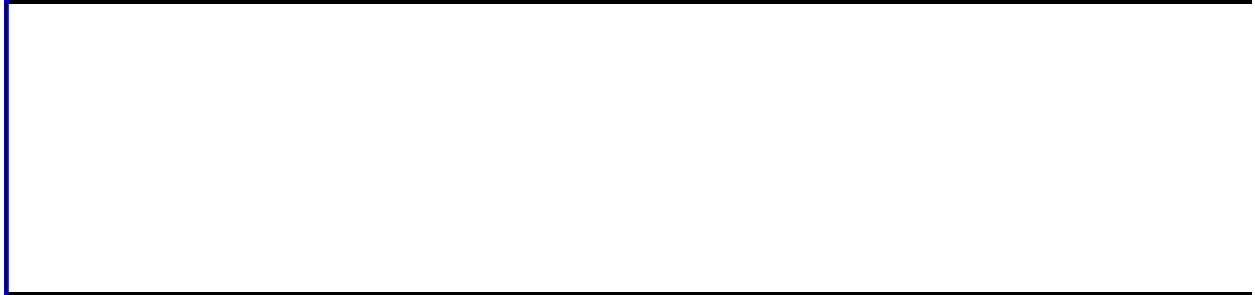
# PivotDetailRange Object

[PivotCell](#) └ [PivotDetailRange](#)  
└ Multiple objects



Represents the range of cells in the detail grid for a PivotTable list.

## Using the PivotDetailRange Object

The [PivotCell](#) object's [DetailRange](#) property returns a **PivotDetailRange** object.



# PivotField Object

Multiple objects  [PivotField](#)  
 Multiple objects

Represents a single field in a PivotTable list.



## Using the PivotField Object

The following properties and methods return a **PivotField** object:

- The [Field](#) property of the following objects: [PivotAxisMember](#), [PivotColumnMember](#), [PivotDetailCell](#), [PivotMember](#), [PivotPageMember](#), [PivotRowMember](#), and [PivotTotal](#).
- The **PivotField** object's [FilterContext](#) property.
- The [PivotFields](#) object's [Item](#) property.
- The [PivotFieldSet](#) object's [AddCalculatedField](#) and [AddCustomGroupField](#) methods, and [BoundField](#) property.
- The [PivotResultField](#) and [PivotResultGroupField](#) objects' [SourceField](#) property.



# PivotFields Collection Object

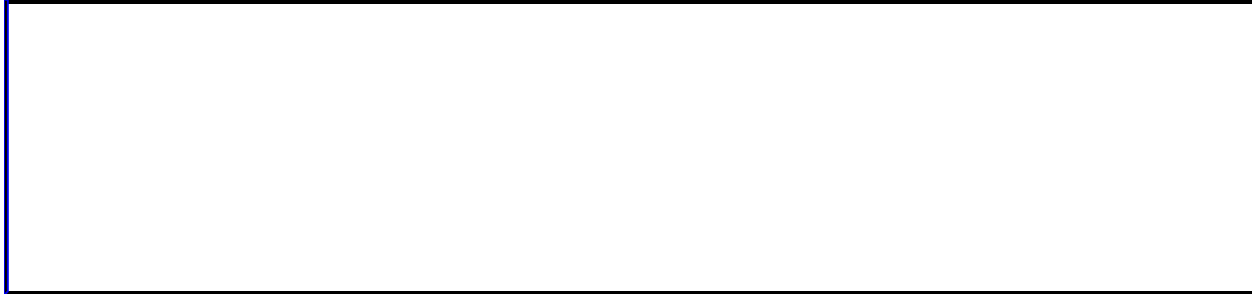
Multiple objects  [PivotFields](#)  
 [PivotField](#)

The collection of [PivotField](#) objects in the specified field set.





## Using the PivotFields Collection

Use the [Fields](#) property of the following objects to return a **PivotFields** collection: [PivotDetailRange](#), [PivotFieldSet](#), [PivotResultColumnAxis](#), [PivotResultDataAxis](#), [PivotResultGroupAxis](#), [PivotResultPageAxis](#), or [PivotResultRowAxis](#).



# PivotFieldSet Object

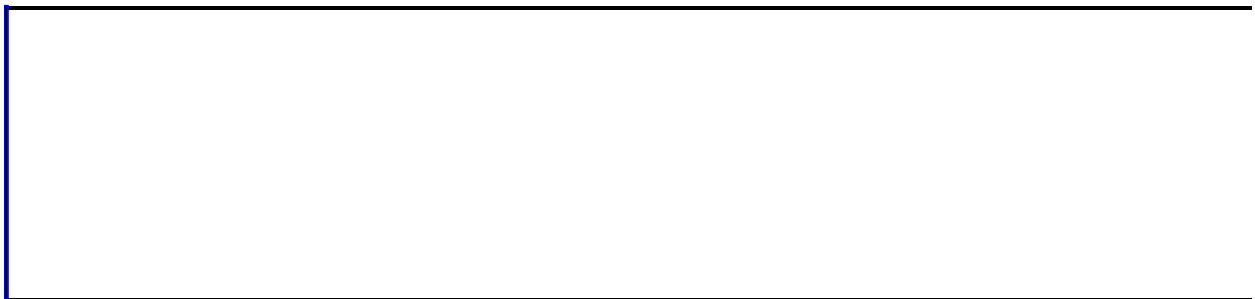
Multiple objects  [PivotFieldSet](#)  
 Multiple objects

Represents a set of fields that have been locked together to form a hierarchy. For example, in a field set for geography, the fields might be Continent, Country/Region, State, and City—in that order. Typically, a field set will only contain a single field if the data source is a recordset.


## Using the PivotFieldSet Object

The following properties and method return a **PivotFieldSet** object:

- The [PivotField](#) object's [FieldSet](#) property.
- The [PivotFieldSets](#) object's [Item](#) property.
- The [PivotView](#) object's [AddFieldSet](#) method.



# PivotFieldSets Collection Object

Multiple objects   
└─ [PivotFieldSet](#)

The collection of [PivotFieldSet](#) objects on the specified axis or in the specified view.



## Using the PivotFieldSets Collection

The following properties return an object from the **PivotFieldSets** collection:

- The [PivotAxis](#) object's [FieldSets](#) property
- The [PivotDataAxis](#) object's **FieldSets** property
- The [PivotFilterAxis](#) object's **FieldSets** property
- The [PivotGroupAxis](#) object's **FieldSets** property
- The [PivotResultFilterAxis](#) object's **FieldSets** property
- The [PivotView](#) object's **FieldSets** property



# PivotFilterAxis Object

Multiple objects  [PivotFilterAxis](#)  
 Multiple objects

Represents the filter axis in a PivotTable list.

## Using the PivotFilterAxis Object

The following properties return a PivotFilterAxis object:

- The [PivotView](#) object's [FilterAxis](#) property
- The [PivotResultFilterAxis](#) object's [SourceFilterAxis](#) property



# PivotFont Object

Multiple objects  [PivotFont](#)

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





## Using the PivotFont Object

The following properties return a **PivotFont** object:

- The [PivotField](#) object's [DetailFont](#), [GroupedFont](#), and [SubtotalFont](#) properties
- The [PivotLabel](#) object's [Font](#) property
- The [PivotView](#) object's [FieldLabelFont](#), [HeaderFont](#), [PropertyCaptionFont](#), [PropertyValueFont](#), and [TotalFont](#) properties



# PivotGroupAxis Object

Multiple objects  [PivotGroupAxis](#)  
 Multiple objects

Represents the group axis in a PivotTable list.


## Using the PivotGroupAxis Object

The following properties return a **PivotGroupAxis** object:

- The [PivotResultColumnAxis](#) object's [SourceColumnAxis](#) property
- The [PivotResultPageAxis](#) object's [SourcePageAxis](#) property
- The [PivotResultRowAxis](#) object's [SourceRowAxis](#) property
- The [PivotView](#) object's [ColumnAxis](#), [PageAxis](#), and [RowAxis](#) properties



# PivotHyperlink Object

Multiple objects  [PivotHyperlink](#)

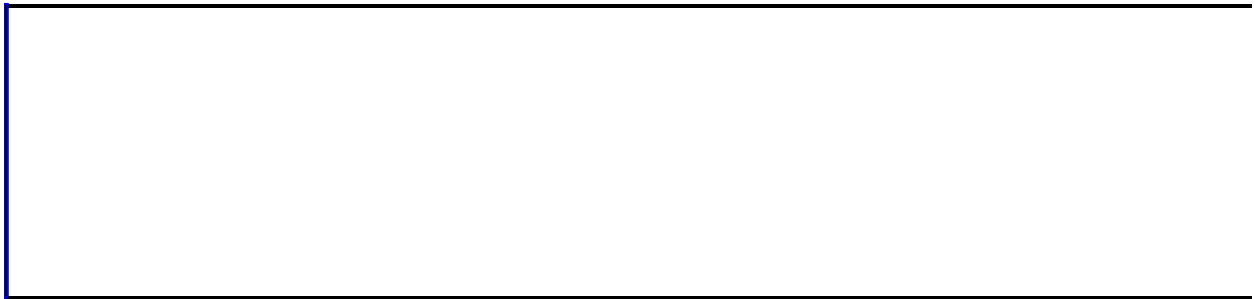
Represents a hyperlink in a PivotTable list.

## Using the PivotHyperlink object



Use the [Hyperlink](#) property of the [PivotDetailCell](#), [PivotAxisMember](#), [PivotColumnMember](#), [PivotRowMember](#) or [PivotPageMember](#) objects to return a **PivotHyperlink** object.

The **PivotTable** control will treat field members as hyperlinks when the [IsHyperlink](#) property of the field is set to **True**.

Use the [Address](#) property to set the address for a **Hyperlink** object. Use the [Follow](#) method to resolve, download, and display the target document.



# PivotLabel Object

Multiple objects  [PivotLabel](#)  
 [PivotFont](#)

Represents the label for the specified object. Contains format attributes (foreground color, background color, font, and so on).



## Using the PivotLabel Object

The following properties return a **PivotLabel** object:

- The [PivotAxis](#) object's [Label](#) property
- The [PivotDataAxis](#) object's **Label** property
- The [PivotFilterAxis](#) object's **Label** property
- The [PivotGroupAxis](#) object's **Label** property
- The [PivotResultLabel](#) object's [SourceLabel](#) property
- The [PivotView](#) object's **Label** property
- The **PivotView** object's [TitleBar](#) property



# PivotMember Object

Multiple objects  [PivotMember](#)  
 Multiple objects

Represents the values displayed for a grouped field.



## Using the PivotMember Object

The following properties and method return a **PivotMember** object:

- The [FindMember](#) property of the following objects: [PivotAxisMember](#), [PivotColumnMember](#), [PivotFieldSet](#), [PivotPageMember](#), or [PivotRowMember](#)
- The [ParentMember](#) property of the following objects: **PivotAxisMember**, **PivotColumnMember**, [PivotMember](#), **PivotPageMember**, or **PivotRowMember**
- The [SourceMember](#) property of the following objects: **PivotAxisMember**, **PivotColumnMember**, **PivotPageMember**, or **PivotRowMember**
- The [PivotField](#) object's [AddCustomGroupMember](#) method
- The [PivotFieldSet](#) object's [AllMember](#) or [Member](#) properties
- The [PivotMembers](#) object's [Item](#) property



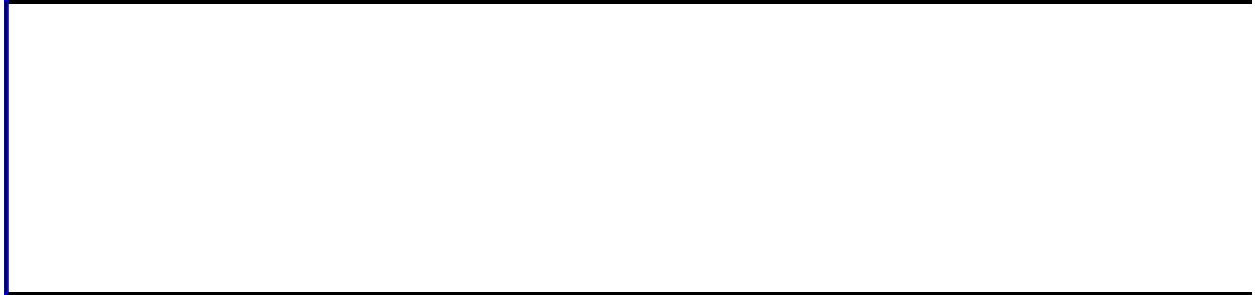
# PivotMemberProperties Collection

[PivotField](#) └ [PivotMemberProperties](#)  
└ [PivotMemberProperty](#)

A collection of [PivotMemberProperty](#) objects.

## Using the PivotMemberProperties collection

Use the [Item](#) property of the **PivotMemberProperties** collection to return a **PivotMemberProperty** object.



# PivotMemberProperty Object

Multiple objects  [PivotMemberProperty](#)

Represents a member property for a PivotTable member. A member property is a custom property that has been defined for the member in an OLAP cube.

## Using the PivotMemberProperty object

The following properties return a **PivotMemberProperty** object:

The [PivotMemberProperties](#) collection's [Item](#) property

The [PivotResultMemberProperty](#) object's [MemberProperty](#) property



Use the [DisplayIn](#) property to control whether the specified member property is displayed in the PivotTable list, ScreenTip, both the PivotTable list and ScreenTip, or not at all. Use the [Caption](#) property to set the caption for a member property.

The following example sets the captions of, and then displays the member captions of the Store Name field.

```
Sub DisplayMemberProperties()  
  
    Dim ptView  
    Dim ptConstants  
    Dim fldStoreName  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view of the PivotTable.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Store Name field.  
    Set fldStoreName = ptView.FieldSets("Store").Fields("Store Name")  
  
    ' The following three lines of code specify that the member prop  
    ' displayed in the PivotTable list.  
    fldStoreName.MemberProperties("Store Manager").DisplayIn = ptCon  
    fldStoreName.MemberProperties("Store Type").DisplayIn = ptConsta  
    fldStoreName.MemberProperties("Store Sqft").DisplayIn = ptConsta  
  
    ' The following three lines of code set the caption for the memb  
    fldStoreName.MemberProperties("Store Manager").Caption = "Manage  
    fldStoreName.MemberProperties("Store Type").Caption = "Store Typ  
    fldStoreName.MemberProperties("Store Sqft").Caption = "Size in S  
  
End Sub
```



# PivotMembers Collection Object

Multiple objects  [PivotMembers](#)  
 [PivotMember](#)

A collection of [PivotMember](#) objects.

## Using the PivotMembers Collection



The following properties return an object from the **PivotMembers** collection:

- The [ChildMembers](#) property of the following objects: [PivotAxisMember](#), [PivotColumnMember](#), [PivotMember](#), [PivotPageMember](#), or [PivotRowMember](#)
- The [PivotField](#) object's [CustomGroupMembers](#) property





# PivotPageMember Object

Multiple objects  [PivotPageMember](#)  
 Multiple objects

Represents the values displayed for a grouped field in the page area of a PivotTable list. The **PivotPageMember** object is a member of the [PivotPageMembers](#) collection.

## Using the PivotPageMember object

The following properties return a **PivotRowMember** object:

The [PivotCell](#) object's [PageMember](#) property

The [PivotResultPageAxis](#) object's **PageMember** property

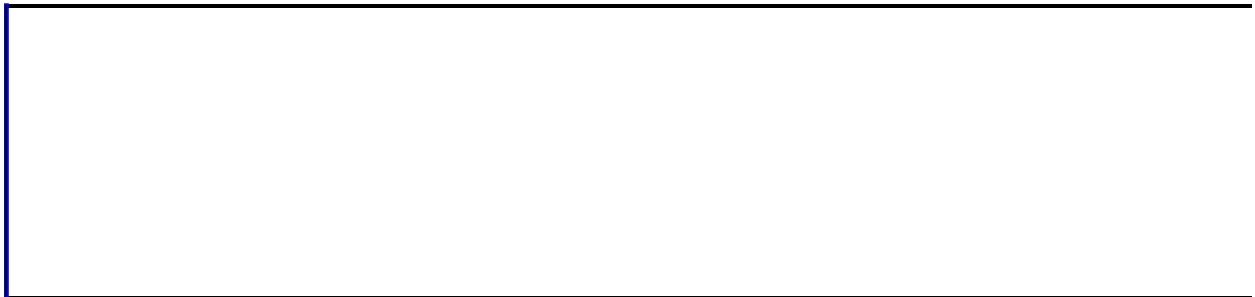
The **PivotPageMember** object's [FindPageMember](#) property

The **PivotPageMember** object's [ParentPageMember](#) property



The **PivotPageMember** object's [TotalPageMember](#) property

The **PivotPageMember** collection's [Item](#) property

The **PivotPageMember** object has many properties in common with the [PivotAxisMember](#) object.



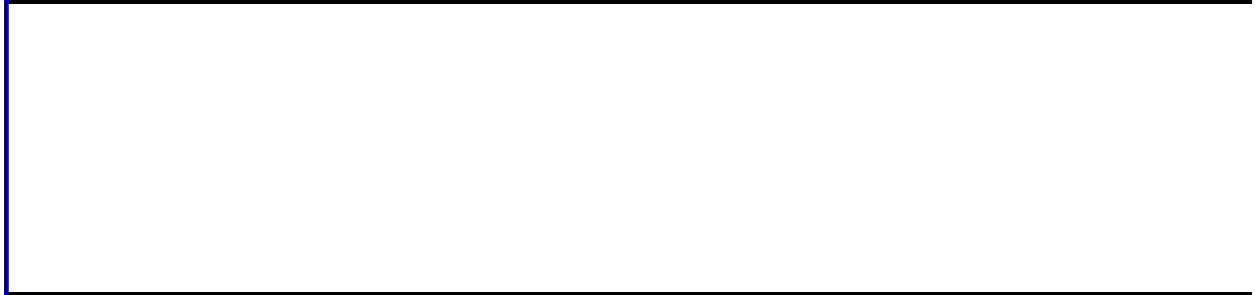
# PivotPageMembers Collection

[PivotPageMember](#)  [PivotPageMembers](#)  
 [PivotPageMember](#)

A collection of [PivotPageMember](#) objects.

## Using the PivotPageMembers Collection

Use the [Item](#) property of the **PivotPageMembers** collection to return a **PivotPageMember** object.



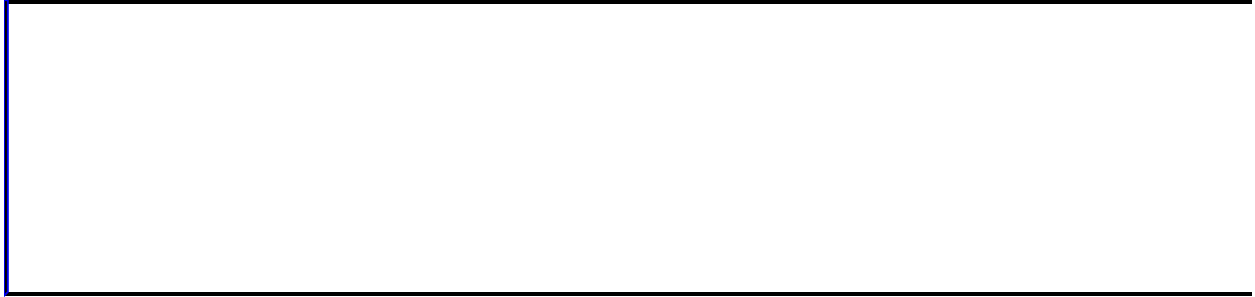
# PivotRange Object

[PivotData](#) └ [PivotRange](#)  
└ Multiple objects



Represents a range of cells in a PivotTable list.

## Using the PivotRange Object

The [PivotData](#) object's [Range](#) property returns a **PivotRange** object.



# PivotResultAxis Object

Multiple objects  [PivotResultAxis](#)  
 Multiple objects

Contains pointers to the data for a result axis in a PivotTable list.

## Using the PivotResultAxis object

The following properties return a **PivotResultAxis** object:

The [PivotResultField](#) object's [Axis](#) property.

The [PivotResultGroupField](#) object's **Axis** property.

Use the [Data](#) property to refer to the data. Use the [SourceAxis](#) property to refer to the source axis.





# PivotResultColumnAxis Object

[PivotData](#) └ [PivotResultColumnAxis](#)  
└ Multiple objects

Represents the data in column axis of a PivotTable list.

## Using the PivotResultColumnAxis object

The [ColumnAxis](#) property of the [PivotData](#) object returns a **PivotResultColumnAxis** object.

Use the [Data](#) property to return a pointer to the data.

Use the [SourceColumnAxis](#) property to return a pointer to the axis.



# PivotResultDataAxis Object

[PivotData](#) └ [PivotResultDataAxis](#)  
└ Multiple objects

Represents the data axis of a PivotTable list.

## Using the PivotResultDataAxis object

The [DataAxis](#) property of the [PivotData](#) object returns a **PivotResultDataAxis** object.

Use the [Data](#) property to return a pointer to the data.

Use the [SourceDataAxis](#) property to return a pointer to the axis.



# PivotResultField Object

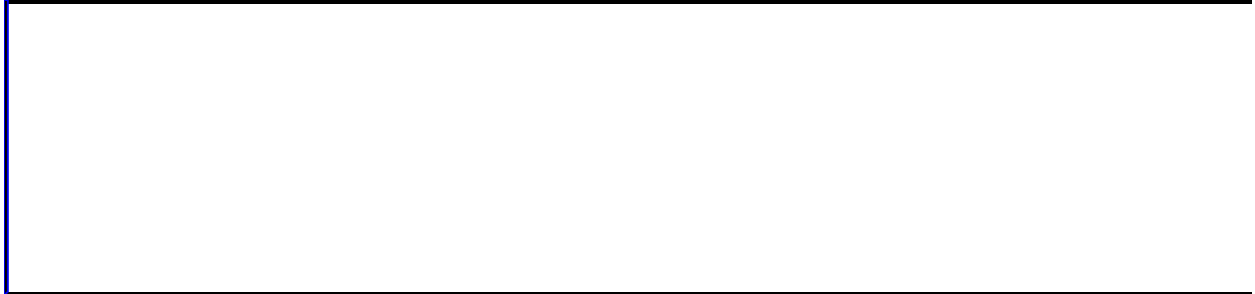
[PivotResultField](#)  Multiple objects

Points to the source field and axis of a result field.

## Using the PivotResultField object

Use the [SourceAxis](#) property of the object returned by the [Axis](#) property to refer to the source axis for the result field.

Use the [SourceField](#) property to refer to the source field for the result field.



# PivotResultFilterAxis Object

[PivotData](#) └ [PivotResultFilterAxis](#)  
└ Multiple objects

Represents the data of the filter axis of a PivotTable list.

## Using the PivotResultFilterAxis object

The [FilterAxis](#) property of the [PivotData](#) object returns a **PivotResultFilterAxis** object.



Use the [Data](#) property to return a pointer to the data.

Use the [SourceFilterAxis](#) property to return a pointer to the axis.





# PivotResultGroupAxis Object

Multiple objects  [PivotResultGroupAxis](#)  
 Multiple objects

Represents the data in the grouping axis of a PivotTable list.

## Using the PivotResultGroupAxis object

The following properties return a **PivotResultGroupAxis** object:

The [ChCategoryLabels](#) object's [PivotAxis](#) property

The [ChSeriesCollection](#) object's **PivotAxis** property

The [PivotAxisMember](#) object's [Axis](#) property

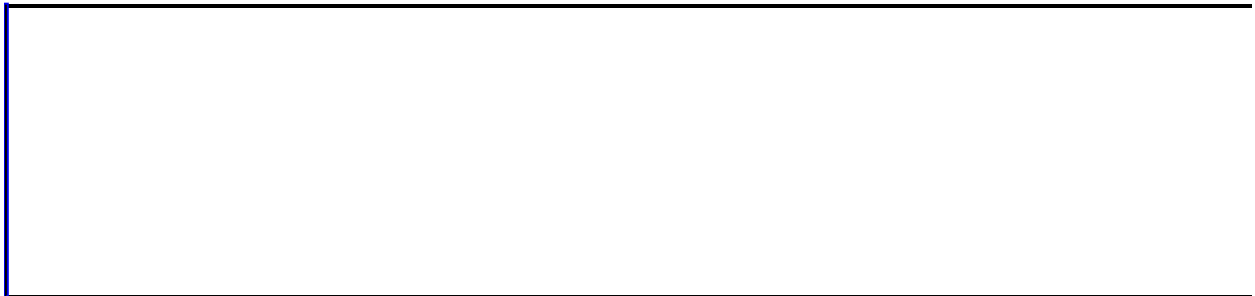
The [PivotColumnMember](#) object's **Axis** property

The [PivotRowMember](#) object's **Axis** property



The [PivotPageMember](#) object's **Axis** property

Use the [Data](#) property to return a pointer to the data.

Use the [SourceAxis](#) property to return a pointer to the axis.



# PivotResultGroupField Object

Multiple objects  [PivotResultGroupField](#)  
 Multiple objects

Represents the grouping field for a result member.

## Using the PivotResultGroupField object

The following properties return a **PivotResultGroupField** object:

The [PivotResultGroupFields](#) collection's [Item](#) property.

The [PivotAxisMember](#) object's [GroupField](#) property.

The [PivotColumnMember](#) object's **GroupField** property.

The [PivotPageMember](#) object's **GroupField** property.

The [PivotRowMember](#) object's **GroupField** property.

Use the [PivotResultAxis](#) object returned by the the [Axis](#) property to configure the source axis.

Use the [PivotField](#) object returned by the [SourceField](#) property to configure the source field.



# PivotResultGroupFields Collection

Multiple objects [└ PivotResultGroupFields](#)  
[└ PivotResultGroupField](#)

A collection of [PivotResultGroupField](#) objects.

## Using the PivotResultGroupFields collection

The following properties return a **PivotResultGroupFields** collection:

The [PivotResultColumnAxis](#) object's [GroupFields](#) property.

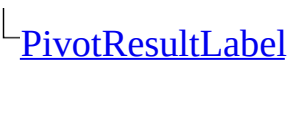
The [PivotResultGroupAxis](#) object's **GroupFields** property.

The [PivotResultPageAxis](#) object's **GroupFields** property.

The [PivotResultRowAxis](#) object's **GroupFields** property.



# PivotResultLabel Object

Multiple objects   
└ [PivotResultLabel](#)  
└ [PivotLabel](#)

Represents the label for a result axis.

## Using the PivotResultLabel object

The following properties return a **PivotResultLabel** object:

The [PivotData](#) object's [Label](#) property

The [PivotResultAxis](#) object's **Label** property

The [PivotResultColumnAxis](#) object's **Label** property

The [PivotResultDataAxis](#) object's **Label** property

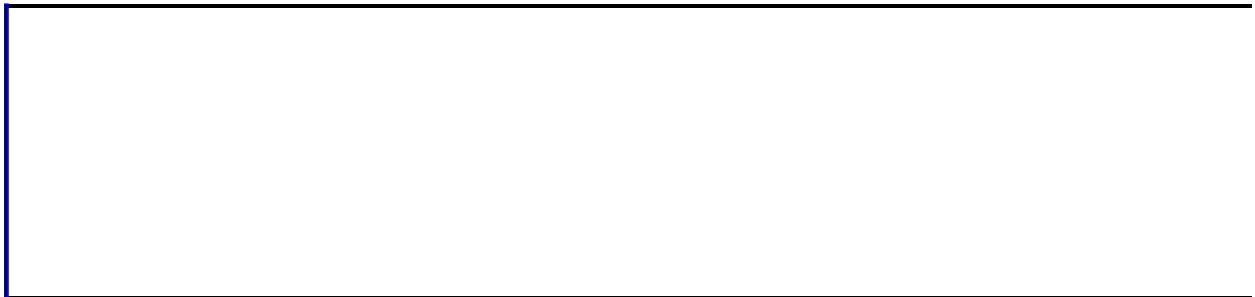
The [PivotResultFilterAxis](#) object's **Label** property

The [PivotResultGroupAxis](#) object's **Label** property

The [PivotResultPageAxis](#) object's **Label** property

The [PivotResultRowAxis](#) object's **Label** property

Use the [PivotLabel](#) object returned by the [SourceLabel](#) property to format the result label.





# PivotResultMemberProperties Collection

Multiple objects  [PivotResultMemberProperties](#)  
 [PivotResultMemberProperty](#)

The collection of member properties for a result member.

## Using the PivotResultMemberProperties collection

The following properties return a **PivotResultMemberProperties** collection:

The [PivotAxisMember](#) object's [MemberProperties](#) property.

The [PivotColumnMember](#) object's **MemberProperties** property.

The [PivotPageMember](#) object's **MemberProperties** property.

The [PivotRowMember](#) object's **MemberProperties** property.



# PivotResultMemberProperty Object

[PivotResultMemberProperties](#) └ [PivotResultMemberProperty](#)  
└ [PivotMemberProperty](#)

Represents a member property for a result member.

## Using the PivotResultMemberProperty object

The [Item](#) property of the [PivotResultMemberProperties](#) collection returns a **PivotResultMemberProperty** object.

Use the [PivotMemberProperty](#) object returned by the [MemberProperty](#) property to access the settings for the member property.



# PivotResultPageAxis Object

[PivotData](#) └ [PivotResultPageAxis](#)  
└ Multiple objects

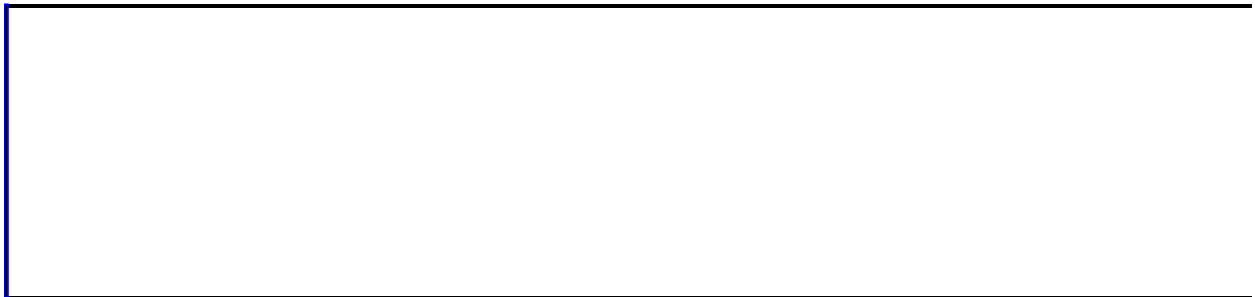
Represents the data the of row axis of a PivotTable list.

## Using the PivotResultPageAxis object

The [PageAxis](#) property of the [PivotData](#) object returns a **PivotResultPageAxis** object.

Use the [Data](#) property to return a pointer to the data.

Use the [SourcePageAxis](#) property to return a pointer to the axis.



# PivotResultRowAxis Object

[PivotData](#) └ [PivotResultRowAxis](#)  
└ Multiple objects

Represents the data the of row axis of a PivotTable list.

## Using the PivotResultRowAxis object

The [RowAxis](#) property of the [PivotData](#) object returns a **PivotResultRowAxis** object.



Use the [Data](#) property to return a pointer to the data.

Use the [SourceRowAxis](#) property to return a pointer to the axis.





# PivotRowMember Object

Multiple objects  [PivotRowMember](#)  
 Multiple objects

Represents the values displayed for a grouped field in the row area of a PivotTable list. The **PivotRowMember** object is a member of the [PivotRowMembers](#) collection.

## Using the PivotRowMember object

The following properties return a **PivotRowMember** object:

The [PivotCell](#) object's [RowMember](#) property.

The [PivotResultRowAxis](#) object's **RowMember** property.

The **PivotRowMember** object's [FindRowMember](#) property.

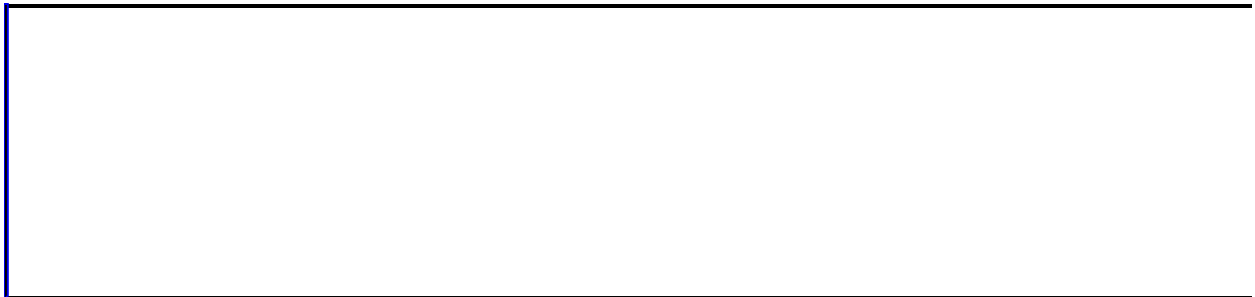
The **PivotRowMember** object's [ParentRowMember](#) property.

The **PivotRowMember** object's [TotalRowMember](#) property.



The **PivotRowMember** collection's [Item](#) property.

The [PivotData](#) object's [Top](#) property.

The **PivotRowMember** object has many properties in common with the [PivotAxisMember](#) object. Use the [TotalRowHeight](#) and [TotalRowMember](#) properties to customize the way totals are displayed.



# PivotRowMembers Collection

Multiple objects  [PivotRowMembers](#)  
 [PivotRowMember](#)

A collection of [PivotRowMember](#) objects.

## Using the PivotRowMembers Collection

Use the [Item](#) property of the **PivotRowMembers** collection to return a **PivotRowMember** object.



# PivotTable Object

[PivotTable](#)   └ Multiple objects

Represents the container for a PivotTable list.

## Using the PivotTable Object



You can use either the **CreateObject** method or the **New** keyword to create a **PivotTable** object.

The object ID for a PivotTable list on an HTML page returns a **PivotTable** object. The programmatic identifier for the **PivotTable** object is CLSID:0002E552-0000-0000-C000-000000000046. The following example creates a PivotTable list named "PivotTable1" on an HTML page.

```
<object id=PivotTable1 classid=CLSID:0002E552-0000-0000-C000-00000000
```



# PivotTotal Object

Multiple objects  [PivotTotal](#)  
 Multiple objects

Represents a total in a PivotTable list. A total is the aggregate value that is displayed for the contents of a given cell.

## Using the PivotTotal Object



The following properties and methods return a **PivotTotal** object:

- The [PivotAggregate](#) object's [Total](#) property
- The [PivotField](#) object's [FilterOn](#) and [SortOn](#) properties
- The [PivotTotals](#) object's [Item](#) property
- The [PivotView](#) object's [AddCalculatedTotal](#) and [AddTotal](#) methods





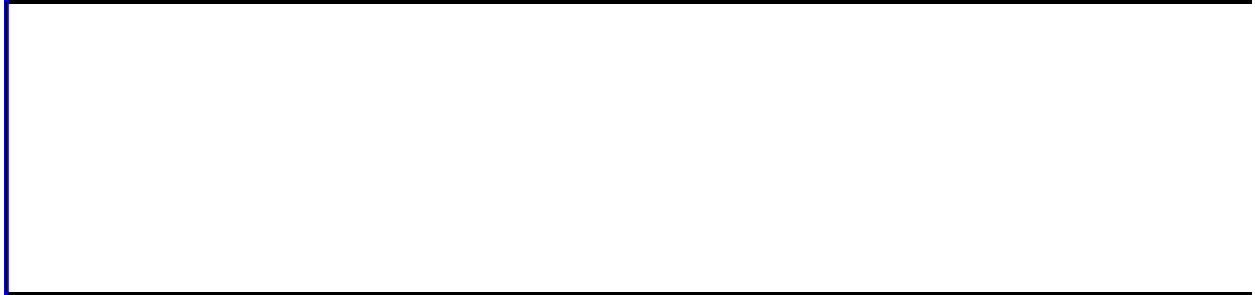
# PivotTotals Collection Object

Multiple objects  [PivotTotals](#)  
 [PivotTotal](#)



The collection of [PivotTotal](#) objects on the specified data axis or in the specified view.

## Using the PivotTotals Collection

The [PivotDataAxis](#), [PivotResultDataAxis](#), and [PivotView](#) objects' [Totals](#) property returns a **PivotTotals** collection.



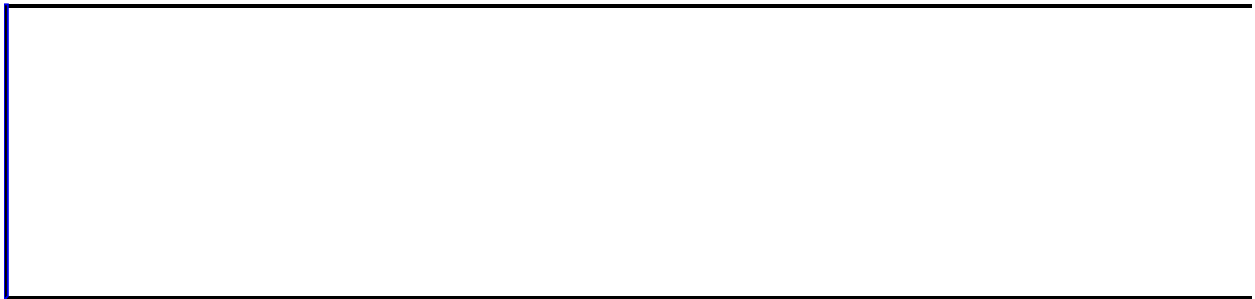
# PivotView Object

Multiple objects  [PivotView](#)  
 Multiple objects


Represents a specific view of a PivotTable list.

## Using the PivotView Object

- The [View](#) property of the following objects returns a **PivotView** object:  
[PivotAxis](#), [PivotAxisMember](#), [PivotColumnMember](#), [PivotData](#),  
[PivotDataAxis](#), [PivotFieldSet](#), [PivotFilterAxis](#), [PivotGroupAxis](#),  
[PivotMember](#), [PivotPageMember](#), [PivotRowMember](#), and [PivotTotal](#).
- The [PivotTable](#) object's [ActiveView](#) property also returns a **PivotView** object.



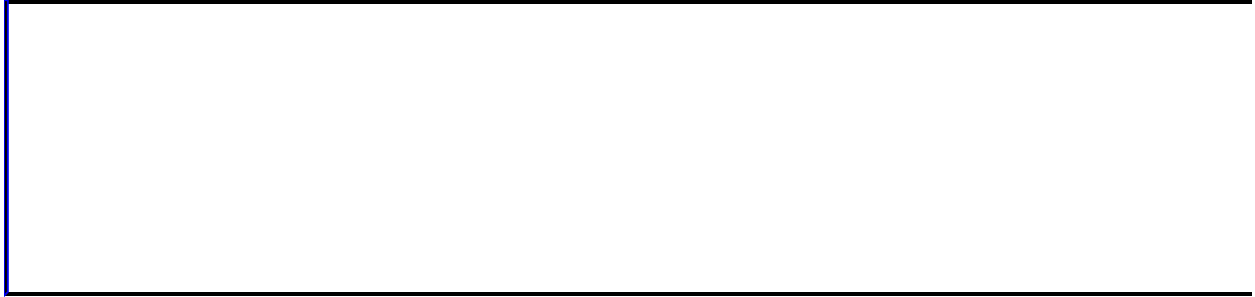
# Protection Object

[Worksheet](#)  [Protection](#)

Contains the protection properties for a worksheet.

## Using the Protection Object

The [Worksheet](#) object's [Protection](#) property returns a **Protection** object.



# Range Object

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

Represents a cell, range of cells, row, or column.

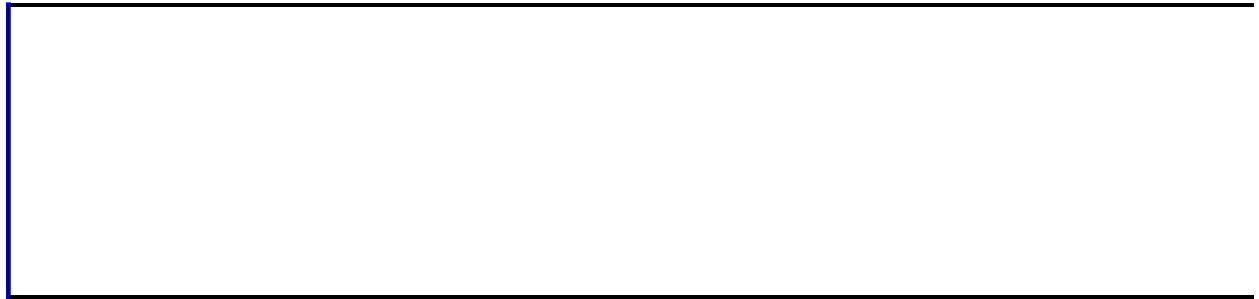
# Using the Range Object

The following properties and methods return a **Range** object.

Object	Properties/Methods
<a href="#"><u>AutoFilter</u></a>	<a href="#"><u>Range</u></a>
<a href="#"><u>Name</u></a>	<a href="#"><u>RefersToRange</u></a>
<a href="#"><u>Pane</u></a>	<a href="#"><u>VisibleRange</u></a>
<a href="#"><u>Range</u></a>	<a href="#"><u>Cells</u></a>
	<a href="#"><u>Columns</u></a>
	<a href="#"><u>CurrentArray</u></a>
	<a href="#"><u>CurrentRegion</u></a>
	<a href="#"><u>End</u></a>
	<a href="#"><u>EntireColumn</u></a>
	<a href="#"><u>EntireRow</u></a>
	<a href="#"><u>Find</u></a>
	<a href="#"><u>FindNext</u></a>
	<a href="#"><u>FindPrevious</u></a>
	<a href="#"><u>Item</u></a>
	<a href="#"><u>MergeArea</u></a>
	<a href="#"><u>Next</u></a>
	<a href="#"><u>Offset</u></a>
	<a href="#"><u>Previous</u></a>
	<a href="#"><u>Range</u></a>
	<a href="#"><u>Rows</u></a>
<a href="#"><u>Spreadsheet</u></a>	<a href="#"><u>ActiveCell</u></a>
	<a href="#"><u>Cells</u></a>
	<a href="#"><u>Columns</u></a>
	<a href="#"><u>Range</u></a>
	<a href="#"><u>RectIntersect</u></a>
	<a href="#"><u>RectUnion</u></a>
	<a href="#"><u>Rows</u></a>



	<a href="#"><u>Selection</u></a>
<a href="#"><u>Window</u></a>	<a href="#"><u>ActiveCell</u></a>
	<a href="#"><u>RangeFromPoint</u></a>
	<a href="#"><u>RangeSelection</u></a>
	<a href="#"><u>Selection</u></a>
	<a href="#"><u>VisibleRange</u></a>
<a href="#"><u>Worksheet</u></a>	<a href="#"><u>Cells</u></a>
	<a href="#"><u>Columns</u></a>
	<a href="#"><u>Range</u></a>
	<a href="#"><u>Rows</u></a>
	<a href="#"><u>UsedRange</u></a>



# RecordNavigationControl Object

[RecordNavigationControl](#)

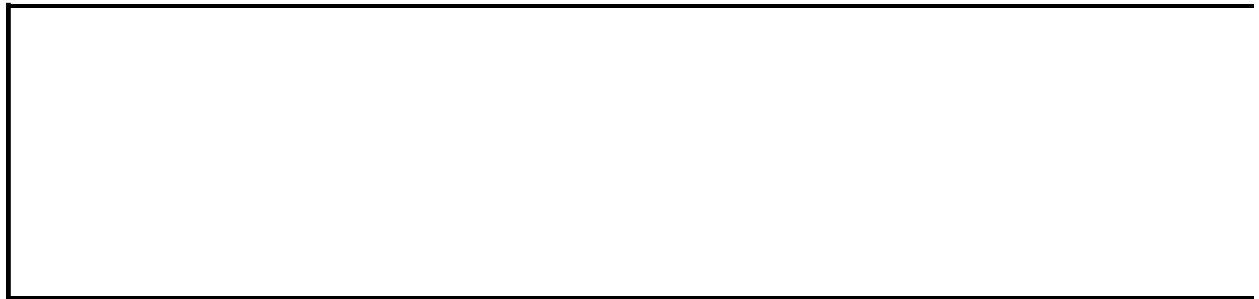
Represents a record navigation control.

## Using the RecordNavigationControl Object



You can use either the **CreateObject** method or the **New** keyword to create a **RecordNavigationControl** object.

The object ID for a record navigation control on an HTML page returns a **RecordNavigationControl** object. The programmatic identifier for the **RecordNavigationControl** object is CLSID:0002E554-0000-0000-C000-000000000046. The following example creates a record navigation control named "RNC1" on an HTML page.

```
<object id=RNC1 classid=CLSID:0002E554-0000-0000-C000-000000000046><
```



# RecordsetDef Object

Multiple objects  [RecordsetDef](#)  
 Multiple objects

Represents a recordset definition.

## Using the RecordsetDef Object

The following properties return a **RecordsetDef** object:

- The [PageField](#) object's [RecordsetDef](#) property
- The [PageRowsource](#) object's [RecordsetDef](#) property
- The **RecordsetDef** object's [ParentRecordsetDef](#) property
- The [RecordsetDefs](#) object's [Item](#) property, and its [Add](#) and [AddNew](#) methods



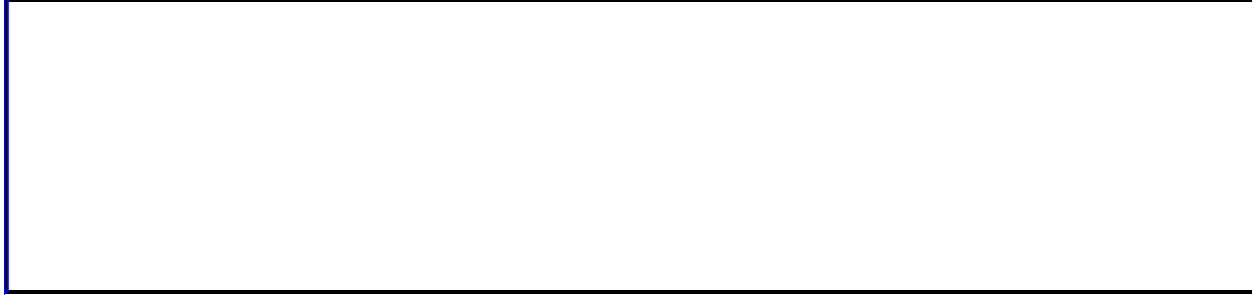
# RecordsetDefs Collection Object

[DataSourceControl](#) └ [RecordsetDefs](#)  
└ [RecordsetDef](#)

The collection of [RecordsetDef](#) objects for the specified data source control.

## Using the RecordsetDefs Collection

The [DataSourceControl](#) object's [RecordsetDefs](#) property return a **RecordsetDefs** collection.



# SchemaField Object

[SchemaFields](#)  [SchemaField](#)

Represents a field in a schema row source. The names of these fields are used in generating SELECT statements for row sources of type **dscTable** or **dscView**. The **SchemaField** object is a member of the **SchemaFields** collection.



## Using the SchemaField Object

The following method and property return a **SchemaField** object:

- The [SchemaFields](#) collection's [Add](#) method
- The **SchemaFields** collection's [Item](#) property



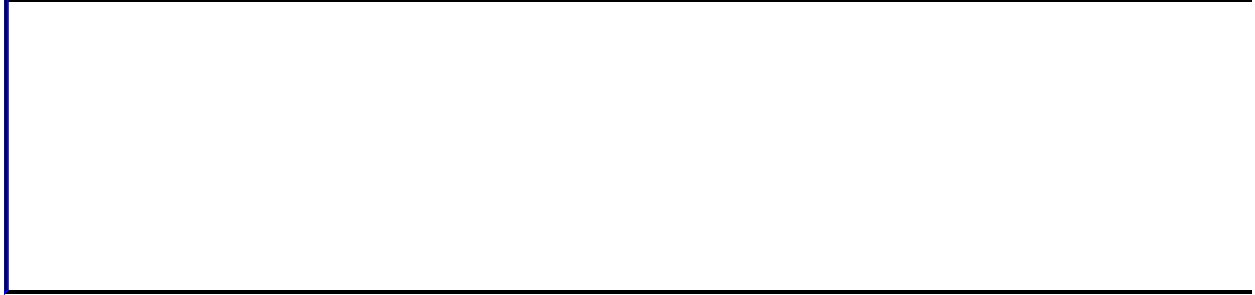
# SchemaFields Collection Object

[SchemaRowsource](#) └ [SchemaFields](#)  
└ [SchemaField](#)

The collection of all available [SchemaField](#) objects in a schema row source.

## Using the SchemaFields Collection

The [SchemaRowsource](#) object's [SchemaFields](#) property returns a **SchemaFields** collection.



# SchemaParameter Object

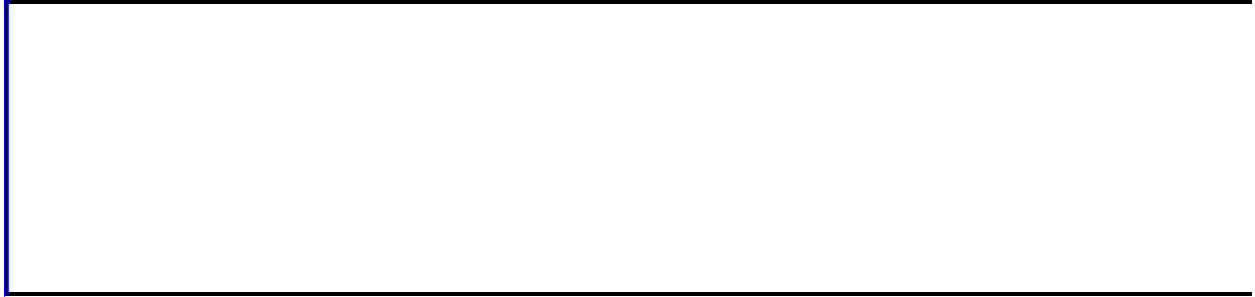
[SchemaParameters](#) └ [SchemaParameter](#)

Represents a single parameter for a schema row source. The **SchemaParameter** object is a member of the **SchemaParameters** collection.

## Using the SchemaParameter Object

The following method and property return a **SchemaParameter** object:

- The [SchemaParameters](#) collection's [Add](#) method
- The **SchemaParameters** collection's [Item](#) property



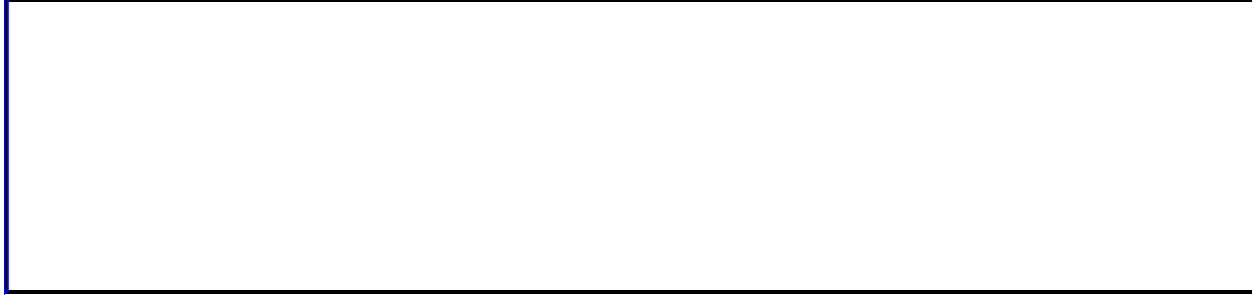
# SchemaParameters Collection Object

[SchemaRowsource](#) └ [SchemaParameters](#)  
└ [SchemaParameter](#)

The collection of [SchemaParameter](#) objects for the specified schema row source.

## Using the SchemaParameters Collection

The [SchemaRowsource](#) object's [SchemaParameters](#) property returns a **SchemaParameters** collection.



# SchemaRelatedField Object

[SchemaRelatedFields](#)  [SchemaRelatedField](#)

Describes the column pairings that make up a schema relationship. The names of these fields are used in generating join clauses in SQL and for tying a parent recordset definition to a sublist child recordset definition. The **SchemaRelatedField** object is a member of the **SchemaRelatedFields** collection.



## Using the SchemaRelatedField

The following method and property return a **SchemaRelatedField** object:

- The [SchemaRelatedFields](#) collection's [Add](#) method
- The **SchemaRelatedFields** collection's [Item](#) property



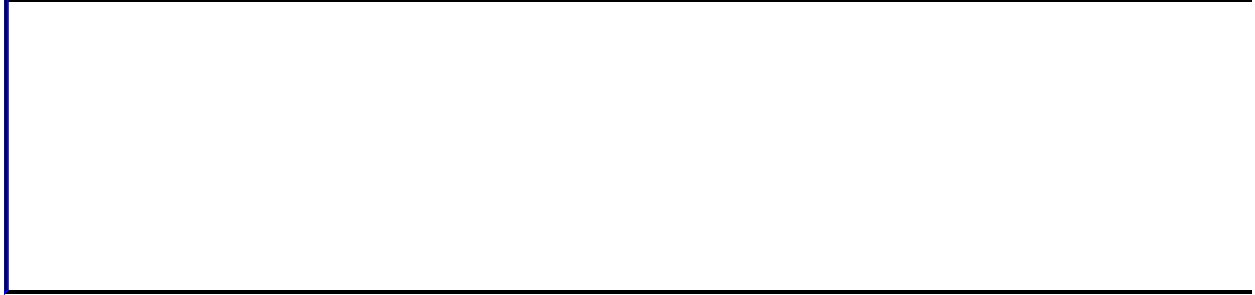
# SchemaRelatedFields Collection Object

[SchemaRelationship](#) └ [SchemaRelatedFields](#)  
└ [SchemaRelatedField](#)



The collection of [SchemaRelatedField](#) objects for a schema relationship.

## Using the SchemaRelatedFields Collection

The [SchemaRelationship](#) object's [SchemaRelatedFields](#) property returns a **SchemaRelatedFields** collection.



# SchemaRelationship Object

Multiple objects  [SchemaRelationship](#)  
 [SchemaRelatedFields](#)

Represents a schema relationship. A schema relationship describes how schema row sources are connected, and it always has a “one” side and a “many” side (from a one-to-many relationship). The **SchemaRelationship** object is a member of the **SchemaRelationships** collection.

## Using the SchemaRelationship Object

The following properties and method return a **SchemaRelationship** object:

- The [LookupSchemaRelationships](#) collection's [Item](#) property
- The [SchemaRelationships](#) collection's [Add](#) and [AddNew](#) methods, and **Item** property
- The [SublistSchemaRelationships](#) collection's **Item** property



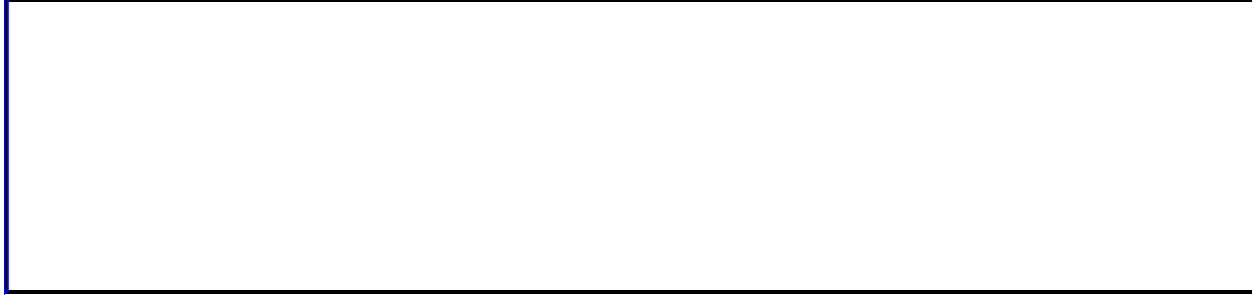
# SchemaRelationships Collection Object

[DataSourceControl](#) └ [SchemaRelationships](#)  
└ [SchemaRelationship](#)

The collection of [SchemaRelationship](#) objects for a data source control.

## Using the SchemaRelationships Collection

The [DataSourceControl](#) object's [SchemaRelationships](#) property returns a **SchemaRelationships** collection.



# SchemaRowsource Object

[SchemaRowsources](#)  [SchemaRowsource](#)

 Multiple objects

Represents a schema row source. Every table, view, or stored procedure in the database is a potential schema row source, as are commands defined explicitly to the data source control.

The **SchemaRowsource** object is a member of the **SchemaRowsources** collection.





## Using the SchemaRowsource Object

The following method and property return a **SchemaRowsource** object:

- The [SchemaRowsources](#) collection's [Add](#) and [AddNew](#) methods
- The **SchemaRowsources** collection's [Item](#) property.



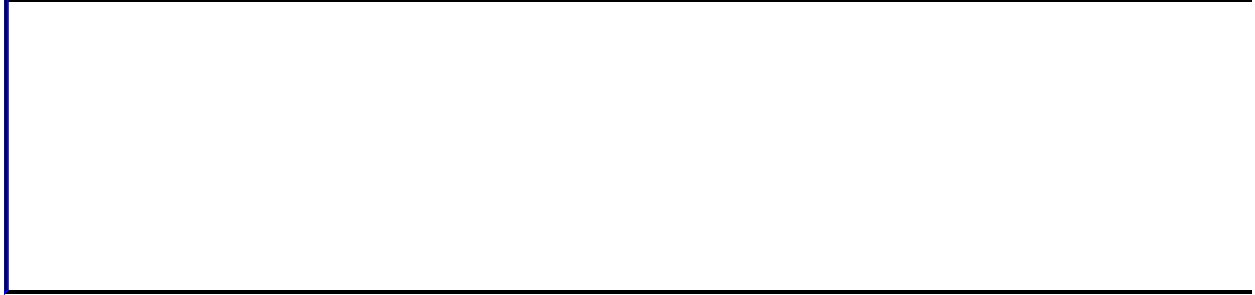
# SchemaRowsources Collection Object

[DataSourceControl](#)  [SchemaRowsources](#)  
 [SchemaRowsource](#)



The collection of **[SchemaRowsource](#)** objects for the specified data source control. This collection is automatically repopulated with objects in the database whenever the database is opened.

## Using the SchemaRowsources Collection

The [DataSourceControl](#) object's [SchemaRowsources](#) property returns a **SchemaRowsources** collection.



# Section Object

Multiple objects  [Section](#)  
 [DataPage](#)

An instance of a single group header, footer, caption, or navigation section.

# Using the Section Object

The following properties and method return a **Section** object:

Object	Properties/Method
<a href="#"><u>DataPage</u></a>	<a href="#"><u>FirstSection</u></a>
<a href="#"><u>DataSourceControl</u></a>	<a href="#"><u>CurrentSection</u></a>
	<a href="#"><u>GetContainingSection</u></a>
<a href="#"><u>DSCEventInfo</u></a>	<a href="#"><u>Section</u></a>
<a href="#"><u>Section</u></a>	<a href="#"><u>ChildSection</u></a>
	<a href="#"><u>NextSection</u></a>
	<a href="#"><u>NextSibling</u></a>
	<a href="#"><u>ParentSection</u></a>
	<a href="#"><u>PreviousSection</u></a>
	<a href="#"><u>PreviousSibling</u></a>

# Sheets Collection

Multiple objects  [Sheets](#)

A collection of all the [Worksheet](#) objects in the workbook. Each **Worksheet** object represents a worksheet.

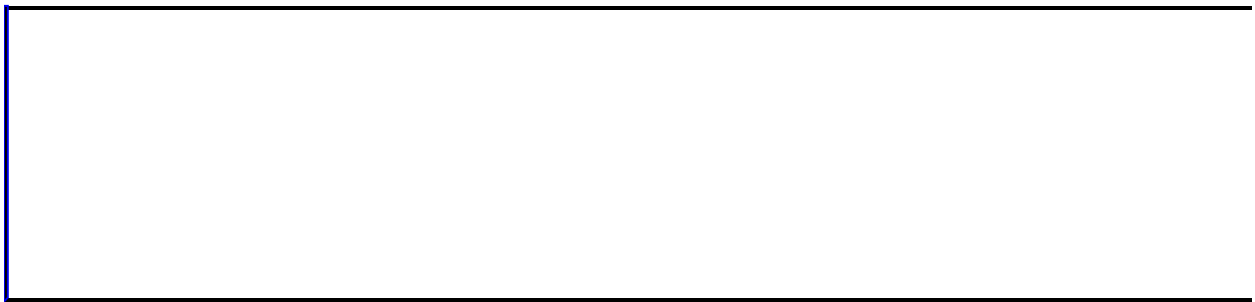
## Using the Sheets collection

The following properties return a **Sheets** collection.

The [Spreadsheet](#) object's [Sheets](#) property.

The [Window](#) object's [SelectedSheets](#) property.

The [Workbook](#) object's **Sheets** property.



# Spreadsheet Object

[Spreadsheet](#) └ Multiple objects

Represents the container for the spreadsheet.



## Using the Spreadsheet Object

You can use the **CreateObject** method or the **New** keyword to create a **Spreadsheet** object.

The object ID for a spreadsheet control on an HTML page or a Visual Basic form returns a **Spreadsheet** object.

The programmatic identifier for the **Spreadsheet** object is CLSID:0002E551-0000-0000-C000-0000000046. The following example creates a spreadsheet named "Spreadsheet1" on an HTML page.

```
<object id=Spreadsheet1 classid=CLSID:0002E551-0000-0000-C000-00000000
```



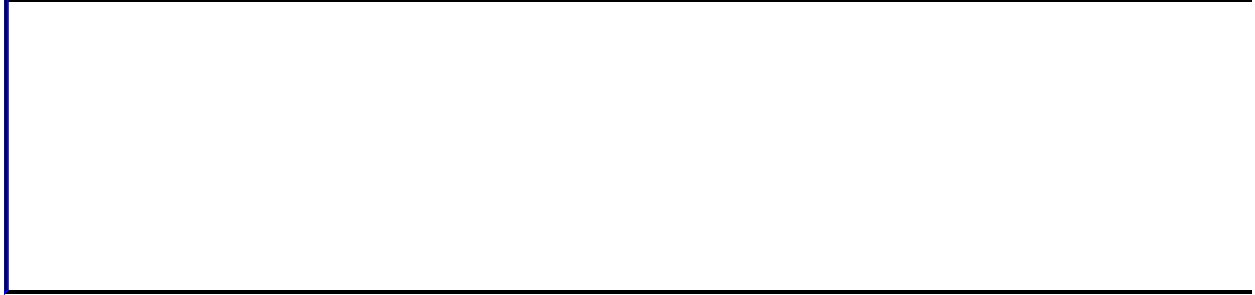
# SublistRelationships Collection Object

[RecordsetDef](#) └ [SublistRelationships](#)  
└ [PageRelationship](#)

The collection of [PageRelationship](#) objects of type **dscSublist** that all have the same recordset definition as their “one” side (parent) of a one-to-many relationship. Records in the child of a sublist relationship are retrieved only when they are needed.

## Using the SublistRelationships Collection

The [RecordsetDef](#) object's [SublistRelationships](#) property returns a **SublistRelationships** collection.



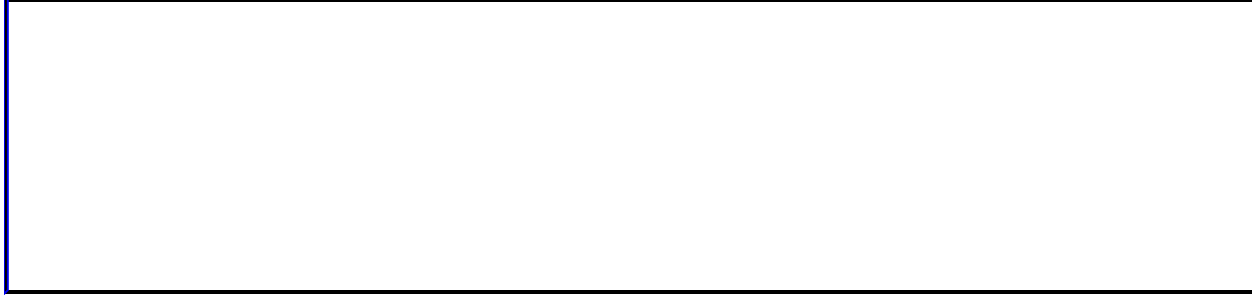
# SublistSchemaRelationships Collection Object

[SchemaRowsource](#) └ [SublistSchemaRelationships](#)  
└ [SchemaRelationship](#)

The collection of [SchemaRelationship](#) objects that all have the same schema row source as their “one” side (from a one-to-many relationship).

## Using the SublistSchemaRelationships Collection

The [SchemaRowsource](#) object's [SublistSchemaRelationships](#) property returns a **SublistSchemaRelationships** collection.



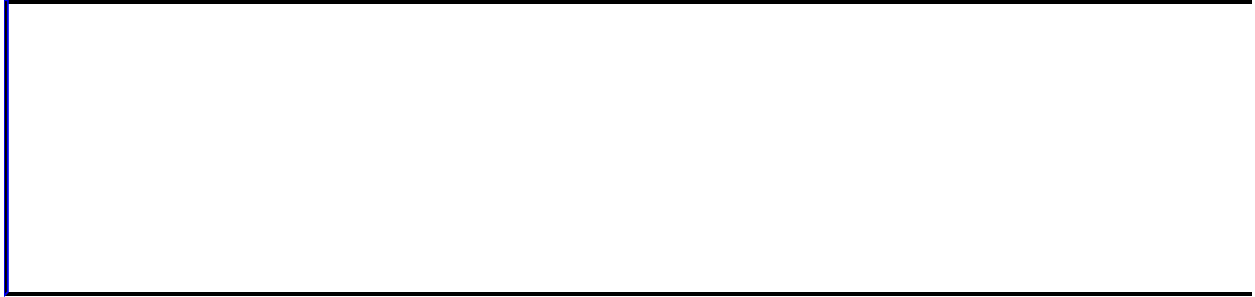
# TitleBar Object

[Spreadsheet](#) └ [TitleBar](#)  
└ Multiple objects



Represents the title bar on the spreadsheet.

## Using the TitleBar Object

The [Spreadsheet](#) object's [TitleBar](#) property returns a **TitleBar** object.



# Window Object

Multiple objects  [Window](#)  
 Multiple objects

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. Each worksheet has a unique **Window** object. Code can only access the window for the active sheet of the workbook. In other words, although the **Windows** collection states that there is only one **Window** object, there are, in fact, multiple **Window** objects, but you may only access the **Window** object for the active sheet.



## Using the Window object

The following properties return a **Window** object.

The **Spreadsheet** object's [ActiveWindow](#) property.

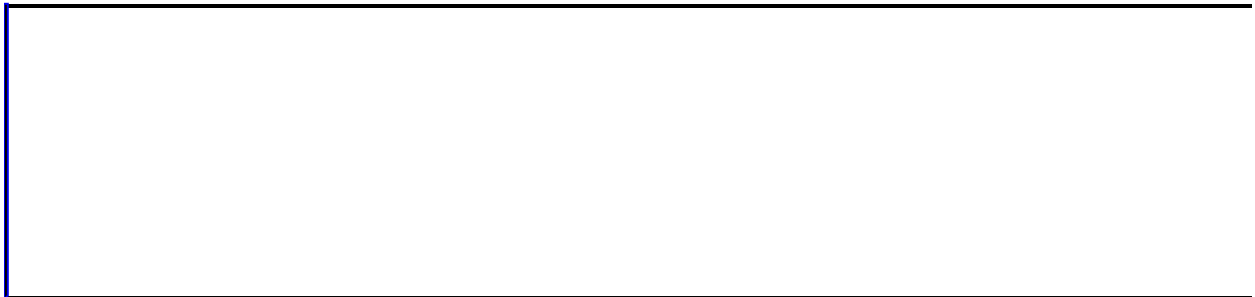
The [Windows](#) collection's [Item](#) property.

The following example hides the row and column headings in the active window of Spreadsheet1.

```
Sub HideHeadings()  
    Spreadsheet1.ActiveWindow.DisplayColumnHeadings = False  
    Spreadsheet1.ActiveWindow.DisplayRowHeadings = False  
End Sub
```

The following example moves column C so that it's the leftmost column in the window.

```
Spreadsheet1.ActiveWindow.ScrollColumn = 3
```



# Windows Collection Object

Multiple objects   
└─ [Windows](#)  
└─ [Window](#)

A collection of all the [Window](#) objects in the Spreadsheet control. The **Windows** collection for the **Spreadsheet** object contains all the windows in the application, whereas the **Windows** collection for the **Workbook** object contains only the windows in the specified workbook. In both cases, the **Spreadsheet** object contains only one **Window** object. This object represents the window for active sheet within the workbook.

Each sheet has a distinct **Window** object associated with it.

`Application.ActiveWindow` or `Application.Workbooks(1).Windows(1)`  
always returns a pointer to the active sheet's window. There is no  
`ActiveSheet.Window` or `Sheets(i).Window`.

## Using the Windows Collection Object



The following properties return a **Windows** collection.

The [Spreadsheet](#) object's [Windows](#) property

The [Workbook](#) object's **Windows** property.



# Workbook Object

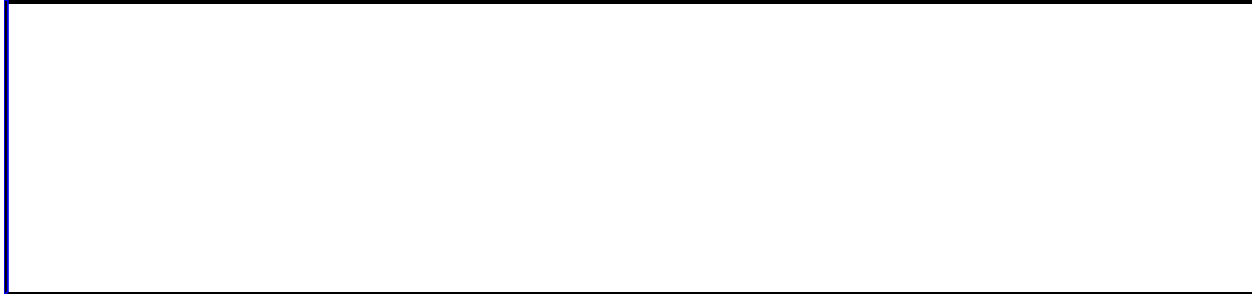
Multiple objects  [Workbook](#)  
 Multiple objects

Represents a workbook. The **Workbook** object is a member of the [Workbooks](#) collection.



## Using the Workbook object

Use the [ActiveWorkbook](#) property of the [Spreadsheet](#) object to return a reference to the open workbook.

A workbook contains a [Worksheets](#) collection.



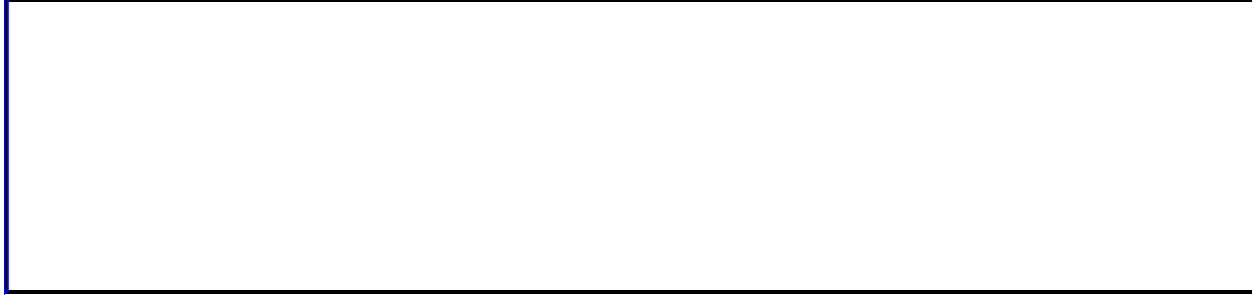
# Workbooks Collection

[Spreadsheet](#)  [Workbooks](#)  
 [Workbook](#)



A collection containing the [Workbook](#) object that is open in the spreadsheet control. The spreadsheet control supports only one open workbook.

## Using the Workbooks collection

Use the [ActiveWorkbook](#) property to refer to the workbook that is currently open in the spreadsheet control.



# Worksheet Object

Multiple objects  [Worksheet](#)  
 Multiple objects

Represents a single worksheet in a workbook.




# Using the Worksheet Object

The following properties return a **Worksheet** object:

- The [Range](#) object's [Worksheet](#) property
- The [Spreadsheet](#), [Window](#), and [Workbook](#) objects' [ActiveSheet](#) property
- The **Worksheet** object's [Next](#) and [Previous](#) properties
- The [Worksheets](#) collection's [Add](#) method and [Item](#) property



# Worksheets Collection Object

Multiple objects   
└─ [Worksheet](#)

A collection of all the [Worksheet](#) objects in the workbook. Each **Worksheet** object represents a worksheet.

## Using the Worksheets Collection Object

Use the [Worksheets](#) property of [Spreadsheet](#) or [Workbook](#) object to return the **Worksheets** collection.

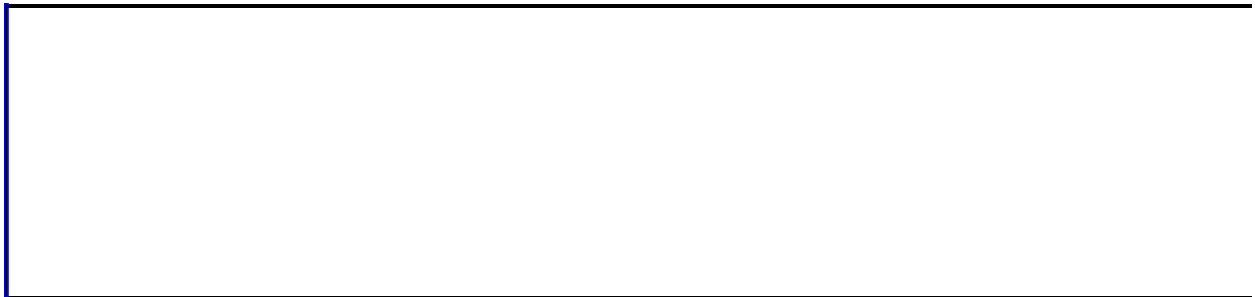
Use the **Add** method to create a new worksheet and add it to the collection. The following example adds two new worksheets before sheet one of Spreadsheet1.

```
Spreadsheet1.Worksheets.Add _  
    Spreadsheet1.Worksheets(1), , 2
```

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 Spreadsheet1.

```
Spreadsheet1.Worksheets(1).Visible = False
```

The **Worksheet** object is also a member of the [Sheets](#) collection.



▼ [Show All](#)

# Activate Method

- ▶ [Activate method as it applies to the \*\*Range\*\* object.](#)

Activates a single cell. To select a range of cells, use the **Select** method.

*expression*.**Activate**

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

- ▶ [Activate method as it applies to the \*\*Worksheet\*\* object.](#)

Activates the specified worksheet.

*expression*.**Activate**

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

# Example

## ► [As it applies to the \*\*Range\*\* object.](#)

This example selects cells A1:C3 on the active worksheet and then makes cell B2 the active cell within the selection.

```
Sub Activate_Range()  
    ' Select cells A1:C3 in the active worksheet.  
    Spreadsheet1.ActiveSheet.Range("A1:C3").Select  
  
    ' Select cell B2. The following line of code uses the Cells  
    ' method to index the current selection. Cell B2 is in the  
    ' second row and the second column of the current selection.  
    Spreadsheet1.Selection.Cells(2, 2).Activate  
End Sub
```

## ► [As it applies to the \*\*Worksheet\*\* object.](#)

This example activates Sheet2 in Spreadsheet1.

```
Spreadsheet1.Worksheets("Sheet2").Activate
```



▼ [Show All](#)

# Add Method

-

## ► [Add method as it applies to the \*\*ChAxes\*\* object.](#)

Adds an axis to a chart. Creates a new **ChAxis** object.

*expression*.**Add(Scaling)**

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

**Scaling** Required **ChScaling** object. A scaling object from another axis or series.

## ► [Add method as it applies to the \*\*ChCharts\*\* object.](#)

Creates a new, empty chart. Returns a **ChChart** object.

*expression*.**Add(Index)**

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

**Index** Optional **Long**. Specifies the position of the new chart. Specifying zero (0) inserts the chart at the beginning of the collection. The default value is -1, which means that if you do not specify this argument, the new chart is inserted at the end of the collection.

## ► [Add method as it applies to the \*\*ChDataLabelsCollection\*\* object.](#)

Adds data labels to the specified series. Creates a new, uninitialized **ChDataLabels** object and adds it to the specified **ChDataLabelsCollection** collection. A series can have only one set of data labels; this method fails if the **ChDataLabelsCollection** collection already contains a **ChDataLabels** object.



*expression*.**Add**

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

► [Add method as it applies to the \*\*ChErrorBarsCollection\*\* object.](#)

Adds error bars to a series. Creates a new, uninitialized **ChErrorBars** object and adds it to the specified **ChErrorBarsCollection** collection. A chart can contain only one set of error bars; this means that an XY (Scatter) chart can contain either x or y error bars, but not both.

*expression*.**Add**

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

► [Add method as it applies to the \*\*ChSegments\*\* object.](#)

Adds a segment to a format map. Returns a **ChSegment** object.

*expression*.**Add**

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

► [Add method as it applies to the \*\*ChSeriesCollection\*\* object.](#)

Adds a series to a chart. Creates a new, empty **ChSeries** object at the specified position in the **ChSeriesCollection** collection.

*expression*.**Add**(*Index*)

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

**Index** Optional **Long**. Specifies the position of the new series in the collection.

Specifying zero (0) places the new series at the beginning of the collection. The default value is -1, which means if you do not specify this argument, the new series is placed at the end of the collection.

► [Add method as it applies to the \*\*ChTrendlines\*\* object.](#)

Adds a trendline to a series. Creates a new, uninitialized **ChTrendline** object and adds it to the end of the specified **ChTrendlines** collection. Each series can contain only one trendline.

*expression*.**Add**

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

► [Add method as it applies to the \*\*ElementExtensions\*\* object.](#)

Creates a new **ElementExtension** object.

*expression*.**Add**(*ElementID*, *FailIfThere*)

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

**ElementID** Required **String**. Specifies the ID tag of an HTML element that will be extended by the new **ElementExtension** object.

**FailIfThere** Optional **Boolean**. If this argument is set to **True** and the object specified by **ElementID** already has an element extension, the **Add** method fails. If this argument is set to **False** (or not specified), the existing element extension is replaced.

► [Add method as it applies to the \*\*GroupingDefs\*\* object.](#)

Creates a new grouping definition. Returns a **GroupingDef** object.

*expression*.**Add**(*GroupingDefName*, *GroupingFieldName*, *PageFieldName*,

## ***Index)***

*expression* Required. An expression that returns a [GroupingDefs](#) object.

***GroupingDefName*** Required **String**. Specifies the name of the new grouping definition.

***GroupingFieldName*** Required **String**. Specifies the name of the grouping field used to create the new definition.

***PageFieldName*** Required **String**. Specifies the name of the page field used to create the new definition.

***Index*** Optional **Variant**. Specifies the ordinal position, the name, or the actual object reference for the grouping definition before which the new grouping definition will be added. If you do not specify this argument, the new grouping definition is added at the end of the collection.

## Remarks

A recordset definition that represents a grouping level has an object type of **GroupingDef**, which is derived from the [RecordsetDef](#) object representing the recordset definition. As far as control binding is concerned, a recordset defined by a **GroupingDef** object behaves just like a recordset defined by a **RecordsetDef** object.

► [Add method as it applies to the GroupLevels object.](#)

Creates a new group level. Returns a [GroupLevel](#) object.

*expression.Add(RecordSource, FailIfThere)*

*expression* Required. An expression that returns a [GroupLevels](#) object.

**RecordSource** Required **String**. Specifies the name of the record source for the new group level.

**FailIfThere** Optional **Boolean**. If this argument is set to **True** and the new group level already exists, the **Add** method fails. The default value is **False**.

► [Add method as it applies to the Sheets and Worksheets objects.](#)

Creates a new worksheet. The new worksheet becomes the active sheet.

*expression.Add(Before, After, Count, Type)*

*expression* Required. An expression that returns one of the above objects.

**Before** Optional **Variant**. An object that specifies the sheet before which the new sheet is added.

**After** Optional **Variant**. 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**.

## Remarks

If **Before** and **After** are both omitted, the new sheet is inserted before the active sheet.

► [Add method as it applies to the \*\*PageFields\*\* object.](#)

Creates a new page field. Returns a [PageField](#) object.

*expression*.**Add**(*Source*, *FieldType*, *Name*, *TotalType*, *Index*)

*expression* Required. An expression that returns one a [PageFields](#) object.

**Source** Required **Variant**. Specifies the source for the new page field. For new fields of type **dscOutput**, this argument specifies a [SchemaField](#) object or the name of a schema field. For fields of type **dscCalculated**, this argument specifies the expression for the calculated field. For new fields of type **dscGrouping**, this argument specifies a **PageField** object or the name of a page field on which this grouping is based

**FieldType** Optional **Variant**. Specifies the new field type. The default value and allowed values depend on the parent object used with this method. For more information, see the “Remarks” section in this topic.

**Name** Optional **Variant**. Specifies the name of the new page field. If you do not specify this argument, the name of an output field is the same as the schema field it is based on, the name of a grouping field is "By<pagefieldname>" or "<agg function>of<pagefieldname>", and the name of a calculated field is "Expr". In all cases, numerals are automatically appended to the new name to make it unique, if necessary.

**TotalType** Optional [DscTotalTypeEnum](#). The type of total.

DscTotalTypeEnum can be one of these DscTotalTypeEnum constants.

**dscAny**

**dscAvg**

**dscCount**

**dscMax**

**dscMin**

**dscNone** *default*

**dscStdev**

**dscSum**

***Index*** Optional **Variant**. Specifies the ordinal position, the name, or the actual object reference for the page field before which the new page field is added. If you do not specify this argument, the new page field is added at the end of the collection. Use this argument only with a [GroupingDef](#) object's **PageFields** collection. For more information, see the “Remarks” section following this paragraph.

## Remarks

The **PageFields** collection has three different parent objects: [RecordsetDef](#), [PageRowsource](#), and **GroupingDef**. You use the same **Add** method for all of these parent objects. There are, however, some differences in defaults and allowed values depending on which parent object you use.

- For **RecordsetDef.PageFields.Add**, the default value for *FieldType* is **dscOutput**. Other allowed *FieldType* values are **dscCalculated** and **dscGrouping**. When *FieldType* is **dscOutput**, the specified source field will be added to the primary page row source in the recordset definition; thus, the source field must exist in the corresponding schema row source. When *FieldType* is **dscGrouping**, the *TotalType* value must be something other than **dscNone** (a nonaggregate grouping field can exist only in a grouping definition, not in a recordset definition). You cannot use the *Index* argument in this scenario.
- For **PageRowsource.PageFields.Add**, the default and only allowed value for *FieldType* is **dscOutput**. The only allowed value for *TotalType* is **dscNone**. You cannot use the *Index* argument in this scenario. Using the **Add** method, you can add output fields to look up row sources.
- For **GroupingDef.PageFields.Add**, the default value for *FieldType* is **dscGrouping**. The other allowed *FieldType* value is **dscCalculated**. You can use the *Index* argument in this scenario (sequence is important in groupings of definitions).

► [Add method as it applies to the \*\*LookupRelationships\*\* and \*\*SublistRelationships\*\* objects.](#)

Creates a new lookup join relationship or a new sublist relationship. Returns a [PageRelationship](#) object.

*expression*.**Add**(*PageRowsource*, *SchemaRelationship*)

*expression* Required. An expression that returns one of the above objects.

*PageRowsource* Required [PageRowsource](#) object. Specifies the page row



source on the “one” side of the one-to-many page relationship.

***SchemaRelationship*** Required [SchemaRelationship](#) object. Describes the connecting fields between the parent page row source on the “many” side of the one-to-many relationship and the page row source specified in this method.

## Remarks

The **SublistRelationships** collection contains **PageRelationship** objects of type **dscSublist** for which a [RecordsetDef](#) object is the “one” side (the parent) of a one-to-many relationship.

► [Add method as it applies to the \*\*ParameterValues\*\* object.](#)

Creates a new parameter value. Returns a [ParameterValue](#) object.

*expression.Add(ParameterName, Value)*

*expression* Required. An expression that returns a [ParameterValues](#) object.

**ParameterName** Required **String**. Specifies the parameter name.

**Value** Optional **Variant**. Specifies the value of the parameter.

► [Add method as it applies to the \*\*RecordsetDefs\*\* object.](#)

Creates a new recordset definition. Returns a [RecordsetDef](#) object.

*expression.Add(SchemaRowsource, Name)*

*expression* Required. An expression that returns a [RecordsetDefs](#) object.

**SchemaRowsource** Required **Variant**. Specifies the schema row source that will be used to create the new recordset definition. This argument can be a [SchemaRowsource](#) object or the name of an existing schema row source.

**Name** Optional **Variant**. Specifies the name of the new recordset definition. If you do not specify this argument, the new recordset definition has the same name as the schema row source specified by the **SchemaRowsource** argument; if necessary, numerals are automatically appended to the new name to make it unique.

## Remarks

Using a single [AddNew](#) method call, you can create a recordset definition, page row source, and schema row source.

► [Add method as it applies to the SchemaFields object.](#)

Creates a new schema field. Returns a [SchemaField](#) object.

*expression*.**Add**(*Name*, *DataType*, *Length*)

*expression* Required. An expression that returns a [SchemaFields](#) object.

**Name** Required **String**. Specifies the name of the new schema field.

**DataType** Required ADO **DataTypeEnum**. Specifies the data type of the new schema field.

**Length** Optional **Variant**. Specifies the length of the new schema field.

## Remarks

In design mode in Microsoft Access, the **SchemaFields** collection is automatically filled with the available schema fields.

► [Add method as it applies to the \*\*SchemaParameters\*\* object.](#)

Creates a new schema parameter. Returns a [SchemaParameter](#) object.

*expression*.**Add**(*Name*, *DataType*, *Size*, *Scale*, *Precision*, *Direction*)

*expression* Required. An expression that returns a [SchemaParameters](#) object.

**Name** Required **String**. Specifies the name of the new schema parameter.

**DataType** Optional **Variant**. Specifies the data type of the new schema parameter. Can be one of the ADO **DataTypeEnum** constants.

**Size** Optional **Variant**. Specifies the maximum length of the new schema parameter.

**Scale** Optional **Variant**. Specifies the maximum number of digits to the right of the decimal point.

**Precision** Optional **Variant**. Specifies the maximum number of digits used for the parameter.

**Direction** Optional **Variant**. Specifies whether the parameter is an input parameter, an output parameter, both an input and output parameter, or a procedure return value. As only input parameters are supported, you should not specify this argument.

## Remarks

In design mode in Microsoft Access, the **SchemaParameters** collection is automatically filled with the available schema parameters.

► [Add method as it applies to the \*\*SchemaRelatedFields\*\* object.](#)

Creates a new related field. Returns a [SchemaRelatedField](#) object.

*expression*.**Add**(*ManySchemaField*, *OneSchemaField*)

*expression* Required. An expression that returns a [SchemaRelatedFields](#) object.

**ManySchemaField** Required **String**. Specifies the name of the schema field on the “one” side of the one-to-many relationship.

**OneSchemaField** Required **String**. Specifies the name of the schema field on the “many” side of the one-to-many relationship.

## Remarks

A **SchemaRelatedField** object describes the column pairings that make up a [SchemaRelationship](#) object. The names of these fields are used in generating join clauses in SQL and parent-child relationships in a hierarchy.

In design mode in Microsoft Access, the [SchemaFields](#) collection is automatically filled with the available schema fields.

► [Add method as it applies to the SchemaRelationships object.](#)

Creates a new schema relationship. Returns a [SchemaRelationship](#) object.

*expression.Add(Name, ManySchemaRowsource, OneSchemaRowsource, ManySchemaField, OneSchemaField)*

*expression* Required. An expression that returns a [SchemaRelationships](#) object.

**Name** Required **String**. Specifies the name of the new schema relationship.

**ManySchemaRowsource** Required **String**. Specifies the name of the row source on the “many” side of this one-to-many relationship.

**OneSchemaRowsource** Required **String**. Specifies the name of the row source on the “one” side of this one-to-many relationship.

**ManySchemaField** Required **String**. Specifies the name of the schema field on the “many” side of this one-to-many relationship.

**OneSchemaField** Required **String**. Specifies the name of the schema field on the “one” side of this one-to-many relationship.

## Remarks

In design mode in Microsoft Access, the **SchemaRelationships** collection is automatically filled with the available schema relationships.

► [Add method as it applies to the \*\*SchemaRowSources\*\* object.](#)

Creates a new schema row source. Returns a [SchemaRowSource](#) object.

*expression.Add(Name, RowSourceType, CommandText)*

*expression* Required. An expression that returns a [SchemaRowSources](#) object.

**Name** Required **String**. Specifies the name of the new row source.

**RowSourceType** Required [DscRowSourceTypeEnum](#). Specifies the type of the new row source.

DscRowSourceTypeEnum can be one of these DscRowSourceTypeEnum constants.

**dscCommandFile**

**dscCommandText**

**dscFunction**

**dscInlineFunction**

**dscProcedure**

**dscTable**

**dscTableFunction**

**dscView**

**CommandText** Optional **Variant**. The schema row source command text. For more information, see the Help topic for the [CommandText](#) property.

## Remarks

In design mode in Microsoft Access, the [SchemaRowsources](#) collection is automatically filled with the available schema row sources.

► [Add method as it applies to the \*\*Criteria\*\* object.](#)

Adds an AutoFilter criterion.

*expression.Add(Criterion)*

*expression* Required. An expression that returns a [Criteria](#) object.

**Criterion** Required **String**. Specifies the new criterion.

► [Add method as it applies to the \*\*Names\*\* object.](#)

Defines a new name. Returns a **Name** object.

*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**. This argument is not supported.



**MacroType** Optional **Variant**. This argument is not supported.

**ShortcutKey** Optional **Variant**. This argument is not supported.

**Category** Optional **Variant**. This argument is not supported.

**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).

## Example

► [As it applies to the \*\*ChCharts\*\* object.](#)

This example adds a chart to the chart workspace.

```
Set cht = ChartSpace1.Charts.Add
```

► [As it applies to the \*\*ChDataLabelsCollection\*\* object.](#)

This example adds data labels to the specified series in the chart workspace.

```
Set ch = ChartSpace1.Charts(0)  
ch.SeriesCollection(0).DataLabelsCollection.Add
```

► [As it applies to the \*\*ChErrorBarsCollection\*\* object.](#)

This example adds error bars to the specified series in the chart workspace.

```
Set eb = ChartSpace1.Charts(0).SeriesCollection(0).ErrorBarsCollecti
```

► [As it applies to the \*\*ChSeriesCollection\*\* object.](#)

This example adds a new series to the specified chart in the chart workspace and then sets the values for the new series.

```
Dim values(7)
```

```
values(0) = 0  
values(1) = 1  
values(2) = 10  
values(3) = 25  
values(4) = 30  
values(5) = 40  
values(6) = 60
```

```
Set s2 = ChartSpace1.Charts(0).SeriesCollection.Add  
Set c = ChartSpace1.Constants  
s2.SetData c.chDimValues, c.chDataLiteral, values
```

► [As it applies to the \*\*ChTrendlines\*\* object.](#)

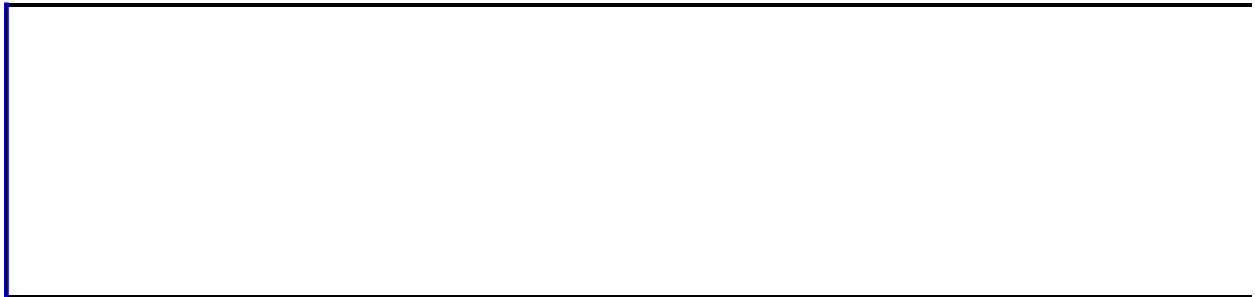
This example adds a trendline to the specified series in the chart workspace.

```
Set trndline = ChartSpace1.Charts(0).SeriesCollection(0).Trendlines.
```

► [As it applies to the \*\*Criteria\*\* object.](#)

This example turns on the AutoFilter for the range A1:J22, and then it sets filters for columns 1 and 3 and applies the filters.

```
Spreadsheet1.Range("a1:j22").AutoFilter  
Set af = Spreadsheet1.ActiveSheet.AutoFilter  
Set filterColumnOne = af.Filters(1)  
Set filterColumnThree = af.Filters(3)  
filterColumnOne.Criteria.Add "blue" ' Don't show blue in column 1.  
filterColumnOne.Criteria.Add "green" ' Don't show green in column 1.  
filterColumnThree.Criteria.Add "yellow" ' Don't show yellow in column 3.  
af.Apply
```



# AddCalculatedField Method

-

Adds a calculated field to a PivotTable. A calculated field can be used like any other field for sorting, filtering, or grouping. You must use the **AddFieldSet** method to create a custom field set before you add a calculated field. Returns a **PivotField** object.

*expression*.**AddCalculatedField**(*Name*, *Caption*, *DataField*, *Expression*)

*expression* Required. An expression that returns a **PivotFieldSet** object that was created by the **AddFieldSet** method.

**Name** Required **String**. The name of the calculated field. The name must be unique within the field set.

**Caption** Required **String**. The caption displayed for the calculated field in the PivotTable user interface. Although this argument is required, you can assign a blank string to the caption.

**DataField** Required **String**. The name of the new field that is created in the underlying recordset for the PivotTable.

**Expression** Required **String**. The expression used to calculate the items in the new field. The expression must be compatible with the Jet expression service.

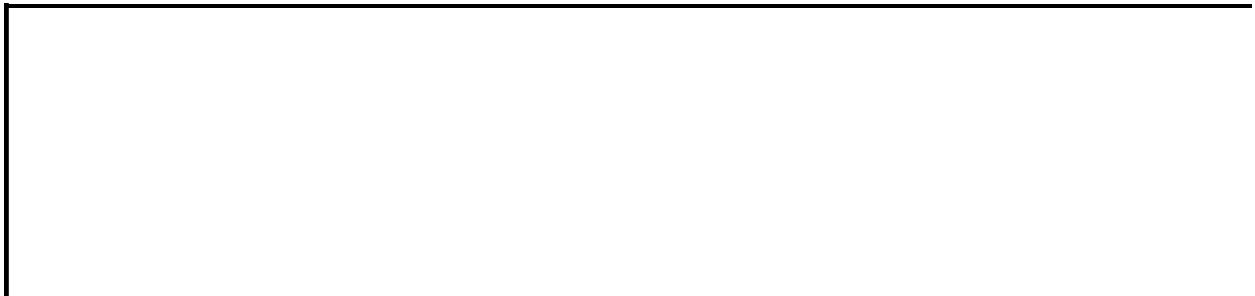
## Remarks

Custom field sets can contain only one calculated field. Adding a second calculated field to a custom field set results in a run-time error.

## Example

This example adds a new field set named "Variance" to PivotTable1. Within the new field set, a calculated field with the caption "Budget Variance" is created. The calculated field is then inserted into the PivotTable view.

```
Sub TestAddFieldSet()  
  
    Dim vwView  
    Dim fsNewFieldSet  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' Add a custom field set to the PivotTable.  
    Set fsNewFieldSet = vwView.AddFieldSet("Variance")  
  
    ' Add a calculated total to the newly created field set.  
    fsNewFieldSet.AddCalculatedField "Variance", _  
    "Budget Variance", "fldVariance", _  
    "Budget / Actual"  
  
    ' Insert the calculated field into the data axis.  
    vwView.DataAxis.InsertFieldSet fsNewFieldSet  
  
End Sub
```



# AddCalculatedTotal Method

-

Adds a calculated total to a PivotTable. Use the **AddCalculatedTotal** method to create a custom total based on totals already defined in the PivotTable. The calculated total is returned as a **PivotTotal** object.

*expression*.**AddCalculatedTotal**(*Name*, *Caption*, *Expression*, *SolveOrder*)

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

**Name** Required **String**. Used to identify the new calculated total in the **PivotTotals** collection. This parameter must be unique within the **PivotTotals** collection. Must be between 1 and 50 characters in length.

**Caption** Required **String**. Used to identify the new calculated total in the PivotTable user interface.

**Expression** Required **String**. The expression used to calculate the new calculated total. Must be a valid multidimensional expression (MDX) statement for the OLE DB provider that is being used to access the data.

**SolveOrder** Optional **Long**. Indicates the solve order of the new calculated total when the PivotTable is refreshed. The SolveOrder parameter is useful if you create a calculated total that is dependent on calculated totals that were created earlier.

## Example

The following example adds a calculated total named "Sales in Last Period" to a PivotTable named "PivotTable1." The new calculated total displays the sales in the previous time period. The example utilizes the FoodMart OLAP cube that is installed with Microsoft SQL Server 7.0 OLAP Services.

```
Sub TestAddCalculatedTotal()  
  
    Dim strExp  
    Dim totCalcTotal  
    Dim vwView  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' The MDX expression used for the new calculated total.  
    strExp = "([Measures].[Store Sales], Time.PrevMember)"  
  
    ' Create the new calculated total.  
    Set totCalcTotal = vwView.AddCalculatedTotal _  
        ("Sales in Last Period", "Sales in Last Period",  
  
    ' Insert the calculated total in the data area of the PivotTable.  
    vwView.DataAxis.InsertTotal totCalcTotal  
  
End Sub
```





# AddCustomGroupField Method

-  
Adds a custom group field to a field set. Returns a **PivotField** object.

*expression*.**AddCustomGroupField**(*Name*, *Caption*, *Before*)

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

***Name*** Optional **String**. The name for the new field.

***Caption*** Optional **String**. The caption to display for the new field.

***Before*** Optional **Variant**. Index, name, or reference to the field.

## Remarks

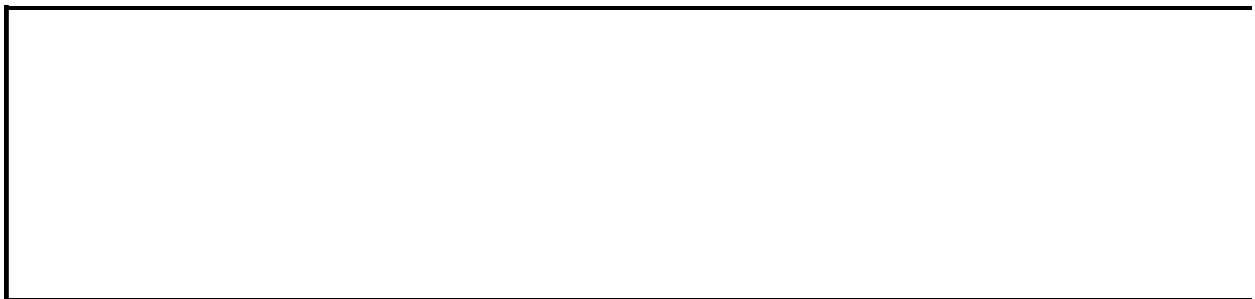
Once you have used this method to create a custom group field, use the **AddCustomGroupMember** method to add members to the group.

Note that the custom group field and its members are created at the client, not the data source.

## Example

This example adds a custom group field to the Time field set, and then adds two members to the field.

```
Sub CreateCustomGroup()  
  
    Dim fsTime  
    Dim fsHalfYear  
  
    ' Set a variable to the Time field set.  
    Set fsTime = PivotTable1.ActiveView.FieldSets("Time")  
  
    ' Add a custom group field named "Group1" to the Time field set.  
    Set fsHalfYear = fsTime.AddCustomGroupField("Group1", "Group1", _  
        "Quarter")  
  
    ' Add a member to the custom field set. This member includes all  
    ' and "Q2" members under 1997.  
    fsHalfYear.AddCustomGroupMember fsTime.Member.Childmembers("1997"  
        Array("Q1","Q2"), "1stHalf")  
  
    ' Add a member to the custom field set. This member includes all  
    ' and "Q4" members under 1997.  
    fsHalfYear.AddCustomGroupMember fsTime.Member.ChildMembers("1997"  
        Array("Q3","Q4"), "2ndHalf")  
  
End Sub
```



# AddCustomGroupMember Method

-

Adds a member to a custom group field. Returns a **PivotMember** object.

*expression*.**AddCustomGroupMember**(*Parent*, *varChildMembers*,  
*bstrCaption*)

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

***Parent*** Required **Variant**. Name, unique name, reference to the parent member for the new member(s).

***varChildMembers*** Required **Variant**. Array of member names, unique names, or member references to add to the new custom group member.

***bstrCaption*** Optional **String**. The caption to display for the new member.

## Remarks

Use this method to populate a field created using the **AddCustomGroupField** method.

## Example

This example adds a custom group field to the Time field set, and then adds two members to the field.

```
Sub CreateCustomGroup()  
  
    Dim fsTime  
    Dim fsHalfYear  
  
    ' Set a variable to the Time field set.  
    Set fsTime = PivotTable1.ActiveView.FieldSets("Time")  
  
    ' Add a custom group field named "Group1" to the Time field set.  
    Set fsHalfYear = fsTime.AddCustomGroupField("Group1", "Group1", _  
        "Quarter")  
  
    ' Add a member to the custom field set. This member includes all  
    ' and "Q2" members under 1997.  
    fsHalfYear.AddCustomGroupMember fsTime.Member.Childmembers("1997"  
        Array("Q1","Q2"), "1stHalf")  
  
    ' Add a member to the custom field set. This member includes all  
    ' and "Q4" under 1997.  
    fsHalfYear.AddCustomGroupMember fsTime.Member.ChildMembers("1997"  
        Array("Q3","Q4"), "2ndHalf")  
  
End Sub
```



# AddFieldSet Method

-

Adds a custom field set to a PivotTable. Once you have created a custom field set, you can use the **AddCalculatedField** method to define a custom field. You cannot add a custom field set to your PivotTable if the PivotTable is connected to an online analytical processing (OLAP) data source. Returns a **PivotFieldSet** object.

*expression*.**AddFieldSet**(*Name*)

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

*Name* Required **String**. Specifies the name of the new field set. The name must be unique within the the **PivotFieldSets** collection. Must be between 1 and 24 characters in length.

## Remarks

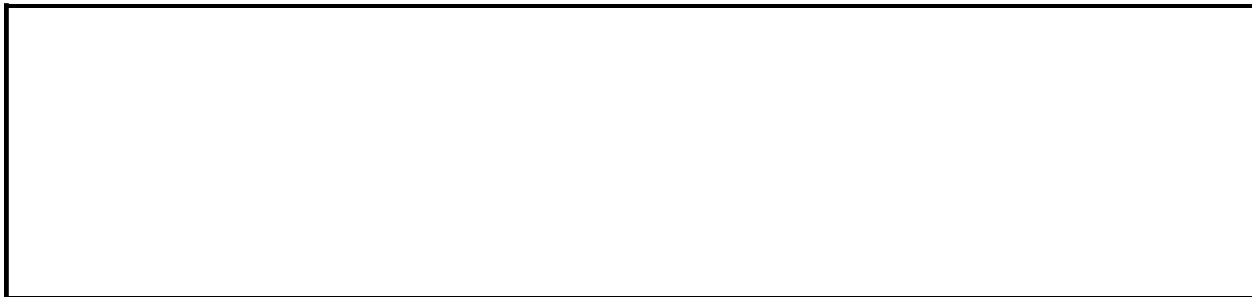
You must add a calculated field to the new field set before you add it to the current PivotTable view. Custom field sets can contain only one calculated field. Adding a second calculated field to a custom field set results in a run-time error.



## Example

This example adds a calculated field named "Variance" to a new field set in PivotTable1. The calculated field is then inserted into the PivotTable view.

```
Sub TestAddFieldSet()  
    Dim vwView  
    Dim fsNewFieldSet  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' Add a custom field set to the PivotTable.  
    Set fsNewFieldSet = vwView.AddFieldSet("Variance")  
  
    ' Add a calculated total to the newly created field set.  
    fsNewFieldSet.AddCalculatedField "Variance", _  
        "Budget Variance", "fldVariance", _  
        "Budget / Actual"  
  
    ' Insert the calculated field into the data axis.  
    vwView.DataAxis.InsertFieldSet fsNewFieldSet  
  
End Sub
```



# AddIn Method

-  
Adds an add-in to the specified spreadsheet.

*expression*.**AddIn**(*AddIn*)

*expression* An expression that returns a [Spreadsheet](#) object.

**AddIn** Required **Object**. Specifies the add-in.

--

# AddNew Method

-

Creates a new schema row source and a recordset definition. Returns a **RecordsetDef** object.

*expression*.**AddNew**(*Source*, *RowSourceType*, *Name*)

*expression* An expression that returns a **RecordsetDefs** object.

**Source** Required **String**. Specifies command text for a new schema row source of type **dscCommandText** or **dscCommandFile**, or for the name of a table, view, or stored procedure to be added as a schema row source. For more information, see the Help topic for the [CommandText](#) property.

**RowSourceType** Optional **Variant**. Specifies the type of the new schema row source. Can be one of the **dscRowSourceTypeEnum** constants. The default constant is **dscCommandText**.

**Name** Optional **Variant**. Specifies the name of the recordset definition, page row source, and schema row source created by using the **AddNew** method. If you do not specify this argument and the value of the **RowSourceType** argument is **dscTable**, **dscView**, or **dscProcedure**, the new name is the same as the name specified in the **Source** argument (with numerals automatically appended to it to make it unique, if necessary). If you do not specify this argument and the value of the **RowSourceType** argument is **dscCommandText** or **dscCommandFile**, the new name is "CommandN", where N is one or more appended numerals.

## Remarks

Page fields are normally created with names that are unique throughout the page data definition. However, when a page field is added to a recordset definition whose primary page row source is of type **dscProcedure**, **dscCommandText**, or **dscCommandFile**, the page field name must be the same as the schema field name. This can result in multiple page field objects with the same name in the **AllPageFields** collection. To differentiate between objects with the same name, you can use the expression

`AllPageFields("RecordsetdefName.PagefieldName")` where

*RecordsetdefName* is the name of the recordset definition, and *PagefieldName* is the name of the page field. Note that page field names themselves cannot contain periods.

▼ [Show All](#)

# AddTotal Method

► [AddTotal method as it applies to the \*\*GroupingDefs\*\* object.](#)

Creates a new total.

*expression*.**AddTotal**(*GroupingDefName*, *GroupingFieldName*, *PageFieldName*, *TotalType*, *Index*)

*expression* An expression that returns a [GroupingDefs](#) object.

**GroupingDefName** Required **String**. Specifies the **GroupingDef** object to be used to create the total.

**GroupingFieldName** Required **String**. Specifies the grouping field to be used to create the total.

**PageFieldName** Required **String**. Specifies the **PageField** object to be used to create the total.

**TotalType** Required [DscTotalTypeEnum](#). Specifies the function to be used to create the total.

DscTotalTypeEnum can be one of these DscTotalTypeEnum constants.

**dscAny**

**dscAvg**

**dscCount**

**dscMax**

**dscMin**

**dscNone**

**dscStdev**

**dscSum**

**Index** Optional **Variant**. Specifies where the new total will be placed in the

collection.

► [AddTotal method as it applies to the PivotView object.](#)

Adds a total to the PivotTable view.

*expression*.**AddTotal**(*Name*, *Field*, *Function*)

*expression* An expression that returns a [PivotView](#) object.

**Name** Required **String**. Specifies the name of the total.

**Field** Required [PivotField](#). Specifies the field to be used to create the total.

**Function** Required [PivotTotalFunctionEnum](#). Specifies the function to be used to create the total.

PivotTotalFunctionEnum can be one of these PivotTotalFunctionEnum constants.

**plFunctionAverage**

**plFunctionCount**

**plFunctionMax**

**plFunctionMin**

**plFunctionStdDev**

**plFunctionStdDevP**

**plFunctionSum**

**plFunctionVar**

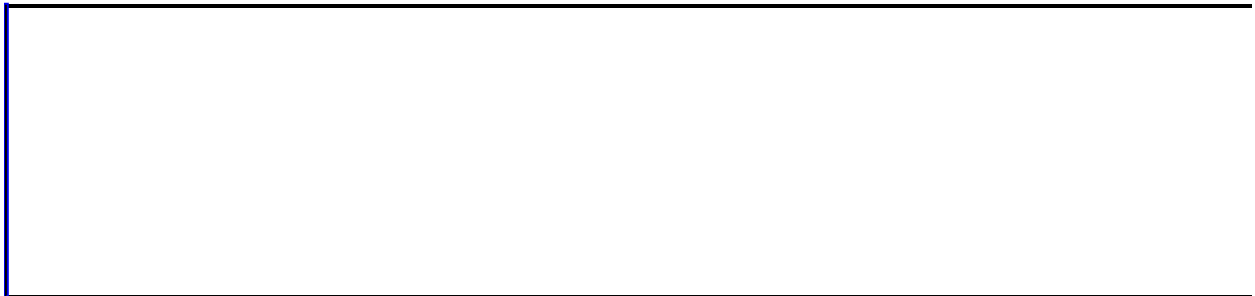
**plFunctionVarP**

## Example

► [As it applies to the \*\*PivotView\*\* object.](#)

This example adds a sum total to the PivotTable list in the active view.

```
Sub Add_Total()  
    Dim vwView  
    Dim ptConstants  
    Dim totNewTotal  
  
    Set vwView = PivotTable1.ActiveView  
    Set ptConstants = PivotTable1.Constants  
  
    ' Add a new total named "Total Budget" to the current view.  
    Set totNewTotal = vwView.AddTotal("Total Budget", vwView.Fieldse  
        ptConstants.plFunctionSum)  
  
    ' Insert the newly created total into the detail area of the Piv  
    vwView.DataAxis.InsertTotal totNewTotal  
  
End Sub
```





# Apply Method

-

Applies the specified AutoFilter. You typically use the [AutoFilter](#) method to turn on the AutoFilter and add one or more criteria to it; you can then use the **Apply** method to apply the new filter.

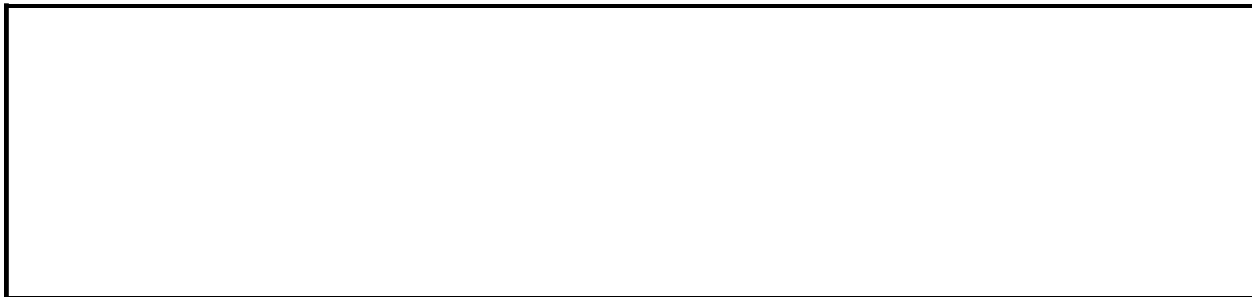
*expression*.**Apply**

*expression* An expression that returns an [AutoFilter](#) object.

## Example

This example turns on the AutoFilter for the range A1:C20, sets filters for columns A and C, and then applies the filters.

```
Private Sub EnableAutoFilter()  
    Dim afFilters  
    Dim afCol1  
    Dim afCol3  
  
    ' Turn on AutoFilter.  
    Spreadsheet1.Worksheets("Sheet1").Range("A1:C20").AutoFilter  
  
    ' Set a variable to the AutoFilter object.  
    Set afFilters = Spreadsheet1.Worksheets("sheet1").AutoFilter  
  
    Set afCol1 = afFilters.Filters(1)  
    Set afCol3 = afFilters.Filters(3)  
  
    ' Add a criteria that excludes blue from column A.  
    afCol1.Criteria.Add "blue"  
  
    ' Add a criteria that excludes green from column A.  
    afCol1.Criteria.Add "green"  
  
    ' Add a criteria that excludes yellow from column c.  
    afCol3.Criteria.Add "yellow"  
  
    ' Apply the criteria.  
    afFilters.Apply  
End Sub
```



# ApplyFilter Method

-

Filters the record on a data access page based upon the currently selected field. Equivalent in functionality to the **Filter by Selection** button on the record navigation control.

*expression*.**ApplyFilter**

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

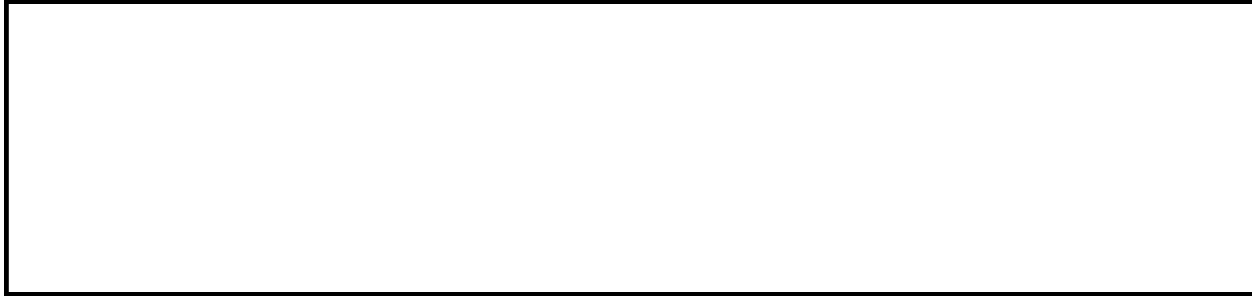
## Remarks

This method relies upon the current selection on the data access page to determine the field to filter by. Therefore, you must set the focus to the field to sort by, when the procedure containing this method is invoked by a control on the data access page, such as a command button.

## Example

This example filters the data access page based upon the currently displayed item in the CategoryID field.

```
Sub Command0_onclick()  
    MSODSC.Datapages(0).FirstSection.HTMLContainer.Children("Category  
    MSODSC.Datapages(0).ApplyFilter  
End Sub
```



# AutoFilter Method

-

Displays or hides the AutoFilter drop-down arrows. You typically use this method to turn on the AutoFilter and add one or more criteria to it; you can then use the [Apply](#) method to apply the new filter.

*expression*.**AutoFilter**

*expression* An expression that returns a [Range](#) object.

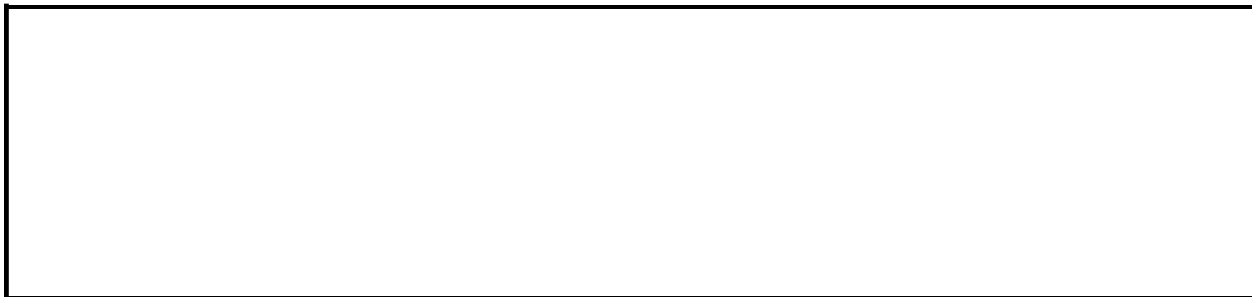
## Remarks

Do not confuse this method with the [AutoFilter](#) property. This method applies to a **Range** object and turns on the AutoFilter, whereas the **AutoFilter** property returns the **AutoFilter** object for a given worksheet.

## Example

This example turns on the AutoFilter for the range A1:C20, sets filters for columns A and C, and then applies the filters.

```
Sub Apply_AutoFilter()  
    Dim afFilters  
    Dim afCol1  
    Dim afCol3  
  
    ' Turn on AutoFilter.  
    Spreadsheet1.Worksheets("Sheet1").Range("A1:C20").AutoFilter  
  
    ' Set a variable to the AutoFilter object.  
    Set afFilters = Spreadsheet1.Worksheets("sheet1").AutoFilter  
  
    Set afCol1 = afFilters.Filters(1)  
    Set afCol3 = afFilters.Filters(3)  
  
    ' Add a criteria that excludes blue from column A.  
    afCol1.Criteria.Add "blue"  
  
    ' Add a criteria that excludes green from column A.  
    afCol1.Criteria.Add "green"  
  
    ' Add a criteria that excludes yellow from column c.  
    afCol3.Criteria.Add "yellow"  
  
    ' Apply the criteria.  
    afFilters.Apply  
End Sub
```





# AutoFit Method

-

Changes the width of the columns in the range or the height of the rows in the range to achieve the best fit.

*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.

## Example

This example adjusts the selected rows and columns to the best fit.

```
Sub AutoFitSelection()  
    Dim rngSelected  
    Set rngSelected = Spreadsheet1.Selection  
    rngSelected.Rows.AutoFit  
    rngSelected.Columns.AutoFit  
End Sub
```



# AutoLayout Method

-

Resets the specified PivotTable list to a default view configuration. If the PivotTable is bound to a recordset, a field set is added to the data axis for each field in the recordset. If the PivotTable is bound to an multidimensional data source, such as an OLAP cube, all field sets are removed from the axes, clearing the current view.

*expression*.**AutoLayout**(*MaxDataFields*)

*expression* An expression that returns a [PivotView](#) object.

**MaxDataFields** Optional **Long**. Specifies the maximum number of fields that can be added to the data axis if the data member is a recordset. If this argument is not specified or is zero (0), there is no limit to the number of fields that will be added.

## Example

This example resets the active view for PivotTable1.

```
PivotTable1.ActiveView.AutoLayout
```



# BeginObject Method

-  
Begins the drawing sequence for the specified **ChChartDraw** object.

*expression*.**BeginObject**(*id*)

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

*id* Required **Long**. Set this argument to a unique identifier that will subsequently be used to identify the object being drawn. Set to -1 to identify the **ChChartDraw** object that has been passed into an event procedure.

## Remarks

You can combine multiple drawing elements into a single user-selectable item by placing them between calls to the **BeginObject** and **EndObject** methods.

## Example

This example uses the BeforeRender event to cancel the drawing of the gridlines and the plot area of the first chart in ChartSpace1. The AfterRenderEvent then replaces the plot area with an ellipse that is drawn after the chart is rendered.

```
Private Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chConstants

    Set chConstants = ChartSpace1.Constants

    ' Check to see if the chart has been rendered.
    If TypeName(chartObject) = "ChChart" Then

        ' The next three lines of code set the interior
        ' and border properties of the ellipse.
        drawObject.Interior.SetPresetGradient _
            chConstants.chGradientHorizontal, _
            chConstants.chGradientVariantStart, _
            Int((24 - 1 + 1) * Rnd + 1)
        drawObject.Border.Weight = 1
        drawObject.Border.Color = "black"

        ' Begin the drawing object.
        drawObject.BeginObject 1

        ' Draw the ellipse.
        drawObject.DrawEllipse chartObject.PlotArea.Left, chartObject.PlotArea.Right, chartObject.PlotArea.Top, chartObject.PlotArea.Bottom

        drawObject.EndObject

    End If

End Sub

Private Sub ChartSpace1_BeforeRender(chartObject, Cancel)

    Select Case TypeName(chartObject)

        Case "ChGridlines"

            ' Cancel the drawing of the gridlines.
            Cancel.Value = True

    End Select

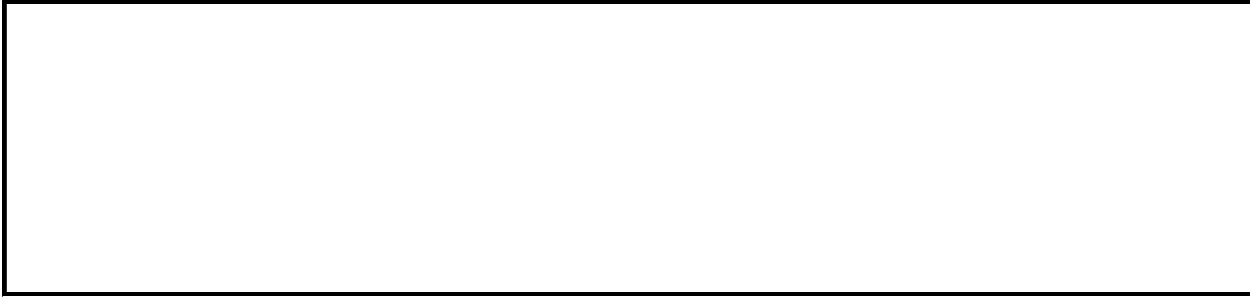
End Sub
```

```
Case "ChPlotArea"
```

```
    ' Cancel the drawing of the plot area.  
    Cancel.Value = True
```

```
End Select
```

```
End Sub
```





# BeginUndo Method

-

Specifies the beginning of an undo block. This means that all statements between this call and its corresponding [EndUndo](#) method call will be undone by a single call to the [Undo](#) method. This makes it possible for you to combine entire macros into one statement that can be easily undone. Undo blocks can be nested.

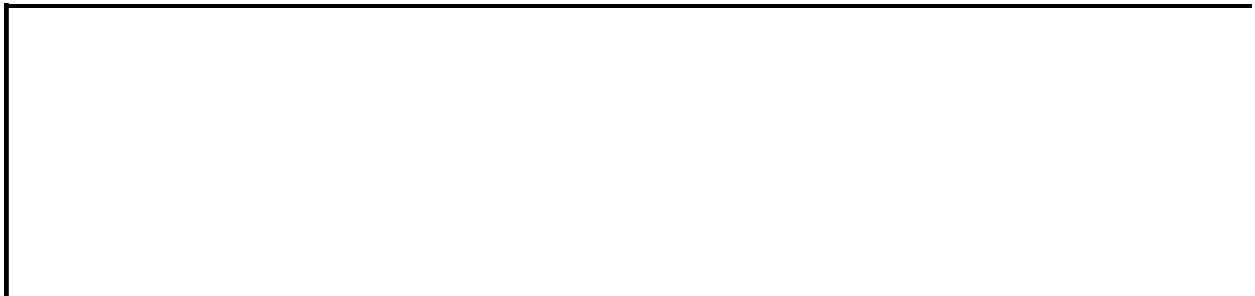
*expression*.**BeginUndo**

*expression* An expression that returns a [ChartSpace](#) or [Spreadsheet](#) object.

## Example

This example creates an undo block containing code that sets the number format and font for cell D10. You can undo all of the formatting by clicking Undo on Spreadsheet1's toolbar.

```
Sub UndoBlock()  
    Dim rngCurrent  
  
    ' Enable undo.  
    Spreadsheet1.EnableUndo = True  
  
    ' Start an undo block.  
    Spreadsheet1.BeginUndo  
  
        Set rngCurrent = Spreadsheet1.Worksheets("sheet1").Range("D10")  
  
        ' The following three lines of code apply  
        ' various formatting to cell D10.  
        rngCurrent.NumberFormat = "0.###"  
        rngCurrent.Font.Color = "Blue"  
        rngCurrent.Font.Name = "Times New Roman"  
  
    ' End the undo block.  
    Spreadsheet1.EndUndo  
End Sub
```



▼ [Show All](#)

# BorderAround Method

-

Adds a border to a range and sets the **Color**, **LineStyle**, and **Weight** properties for the new border.

*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 a [XLLineStyle](#) constant.

XLLineStyle can be one of these XLLineStyle constants.

**xlContinuous** *default*

**xlDash**

**xlDashDot**

**xlDashDotDot**

**xlDot**

**Weight** Optional [XlBorderWeight](#). The border weight.

XlBorderWeight can be one of these XlBorderWeight constants.

**xlHairline**

**xlMedium**

**xlThick**

**xlThin** *default*

**ColorIndex** Optional [XlColorIndex](#). The border color, as an index into the current color palette, or as an XlColorIndex constant.

XlColorIndex can be one of these XlColorIndex constants.

**xlColorIndexAutomatic** *default*

**xlColorIndexNone**

***Color*** Optional **Variant**. The border color, as an RGB value.

## Remarks

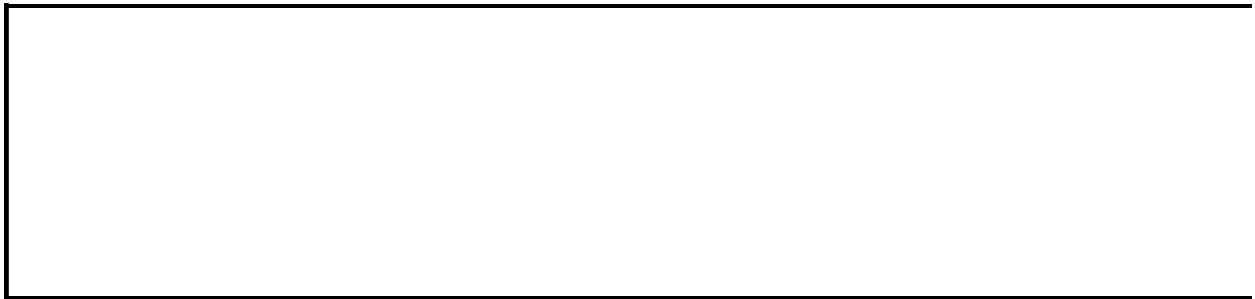
You must specify either ***ColorIndex*** or ***Color***, but not both.

You can specify either ***LineStyle*** or ***Weight***, but not both. If you don't specify either argument, the default line style and weight are used.

## Example

This example adds a thick red border around the range A1:D4 on Sheet1.

```
Sub Add_Border()  
    Dim ssConstants  
    Set ssConstants = Spreadsheet1.Constants  
    Spreadsheet1.Worksheets("Sheet1").Range("A1:D4") _  
        .BorderAround , ssConstants.xlThick, 3  
End Sub
```



# Calculate Method

-

Calculates the open workbook, a specific worksheet in a workbook, or a specified range of cells on a worksheet.

*expression*.**Calculate**

*expression* An expression that returns a **Range**, **Spreadsheet**, or **Worksheet** object.



## Example

This example causes the active worksheet on the spreadsheet to be recalculated.

`Spreadsheet1.ActiveSheet.Calculate`

This example causes the range A1:G5 in Sheet1 to be recalculated.

`Spreadsheet1.Worksheets("Sheet1").Range("A1:G5").Calculate`



# CalculateFull Method

-

Forces every formula in the open workbook to be recalculated.

*expression*.**CalculateFull**

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

## Example

This example forces a full calculation of all data in Spreadsheet1.

Spreadsheet1.**CalculateFull**



▼ [Show All](#)

# Clear Method

-

- ▶ [Clear method as it applies to the \*\*ChartSpace\*\* object.](#)

Deletes every chart in the chart workspace and resets all the formatting to the default values.

*expression*.**Clear**

*expression* An expression that returns a **ChartSpace** object.

- ▶ [Clear Method as it applies to the \*\*Range\*\* object.](#)

Deletes all formatting and data from the specified range.

*expression*.**Clear**

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

## Example

- ▶ [As it applies to the \*\*ChartSpace\*\* object.](#)

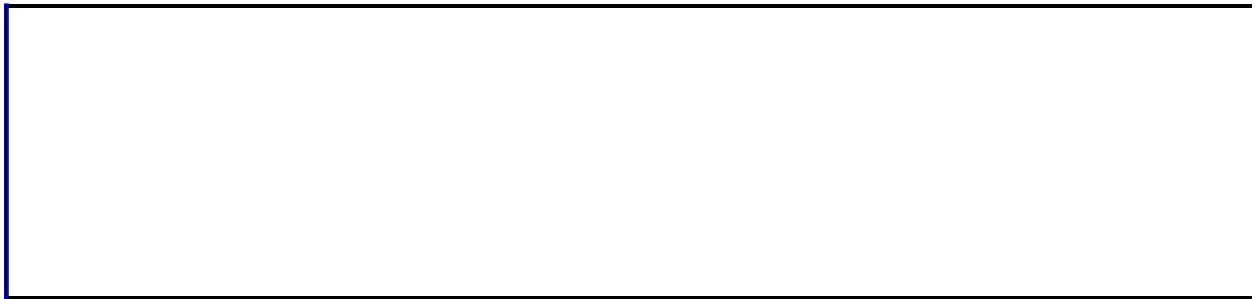
This example deletes all charts in the specified chart workspace.

```
ChartSpace1.Clear
```

- ▶ [At is applies to the \*\*Range\*\* object.](#)

This example clears the range B2:C4 on the active worksheet in Spreadsheet1.

```
Spreadsheet1.ActiveSheet.Range("B2:C4").Clear
```



# ClearContents Method

-

Deletes all data from the specified range.

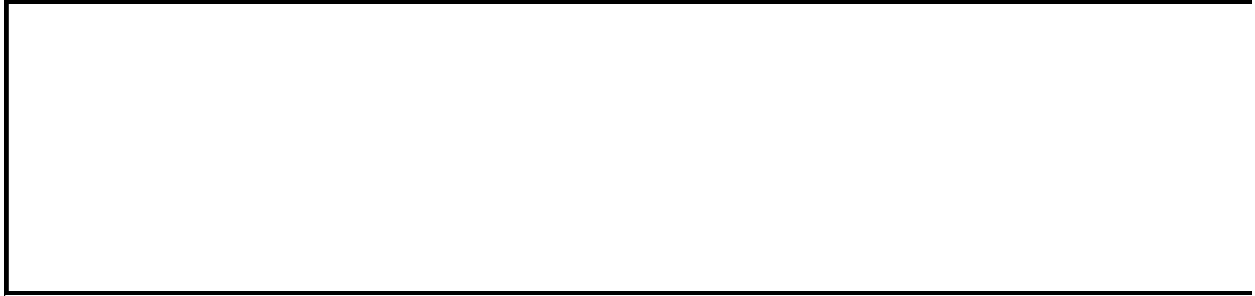
*expression*.**ClearContents**

*expression* An expression that returns a [Range](#) object.

## Example

This example deletes all data from cells A4:B10 on the active worksheet in Spreadsheet1.

```
Spreadsheet1.ActiveSheet.Range("A4:B10").ClearContents
```





# ClearFormats Method

-

Deletes all formatting from the specified range.

*expression*.**ClearFormats**

*expression* An expression that returns a [Range](#) object.

## Example

This example clears the formatting from all cells on the active worksheet.

```
Spreadsheet1.ActiveSheet.Cells.ClearFormats
```



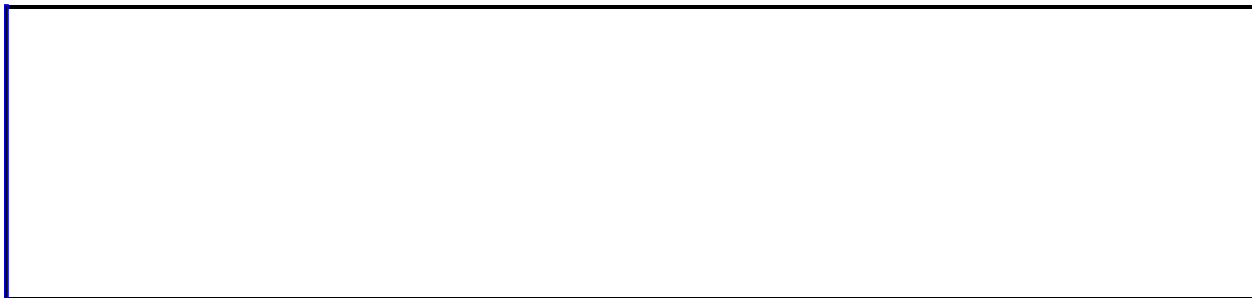
# Collapse Method

-

Causes the specified section to collapse.

*expression*.**Collapse**

*expression* An expression that returns a [Section](#) object.



▼ [Show All](#)

# Copy Method

- ▶ [Copy method as it applies to the \*\*Sheets\*\*, \*\*Worksheet\*\*, and \*\*Worksheets\*\* objects.](#)

Copies the specified sheet to another location in the workbook.

*expression*.**Copy**(*Before*, *After*)

*expression* Required. An expression that returns one of the above objects.

**Before** 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 before which the copied sheet will be placed. You cannot specify **Before** if you specify **After**.

- ▶ [Copy method as it applies to the \*\*Range\*\* object.](#)

Copies the range to the cell location specified in the **Destination** argument. If you omit the **Destination** argument, the range is copied to the Clipboard.

*expression*.**Copy**(*Destination*)

*expression* Required. An expression that returns one of the above objects.

**Destination** Optional **Variant**. Specifies the new range to which the specified range will be copied.

- ▶ [Copy method as it applies to the \*\*PivotTable\*\* object.](#)

Copies the **PivotTable** object to the Windows Clipboard.

*expression*.**Copy**(*Selection*)

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

*Selection* Optional **Object**. The specified selection.

## Example

► [As it applies to the \*\*Sheets\*\*, \*\*Worksheet\*\*, and \*\*Sheets\*\* objects.](#)

This example makes a copy of Sheet1 so that it appears at the end of the worksheet list.

```
Sub CopySheet()  
    ' Copy Sheet1.  
    Spreadsheet1.Sheets("Sheet1").Copy , _  
        Spreadsheet1.Sheets(Spreadsheet1.Sheets.Count)  
  
    ' Rename the new copy of Sheet1.  
    Spreadsheet1.ActiveSheet.Name = "Copy of Sheet1"  
  
End Sub
```

► [As it applies to the \*\*Range\*\* object.](#)

This example copies cells A1:B10 of Sheet1 to a range beginning at the first blank cell in column A of Sheet2.

```
Sub CopyCells()  
    Dim ssConstants  
    Dim rngDest  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Set a variable to the first blank cell in column A of Sheet2.  
    Set rngDest = Spreadsheet1.Sheets("Sheet2").Range("A262144").End(  
        xlUp)  
  
    ' Copy cell2 A1:B10 of Sheet1 to the first blank cell in column A  
    Spreadsheet1.Sheets("Sheet1").Range("A1:B10").Copy rngDest  
  
End Sub
```

►

[As it applies to the \*\*PivotTable\*\* object.](#)

This example copies the active view of PivotTable1 to a new Microsoft Excel worksheet, prints the worksheet, and then closes Excel.

```
Sub Copy_To_XL()
```

```
Dim xlApp
Dim xlBook

' Create a new instance of Excel.
Set xlApp = CreateObject("Excel.Application.10")

' Create a new workbook.
Set xlBook = xlApp.Workbooks.Add

' Copy the current view of the PivotTable.
PivotTable1.Copy PivotTable1.ActiveView

' Paste the PivotTable.
xlApp.ActiveSheet.Paste
' Format the columns.
xlApp.Selection.Columns.AutoFit
' Print the PivotTable.
xlApp.ActiveWindow.SelectedSheets.PrintOut 1

' Close the workbook.
xlBook.Close False

' Exit Excel.
xlApp.Quit

' Clean up variables.
Set xlBook = Nothing
Set xlApp = Nothing
End Sub
```



# CopyFromRecordset Method

-

Copies the contents of an ADO or 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.

*expression*.**CopyFromRecordset**(*Data*, *MaxRows*, *MaxColumns*)

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

**Data** Required **Variant**. The name of 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

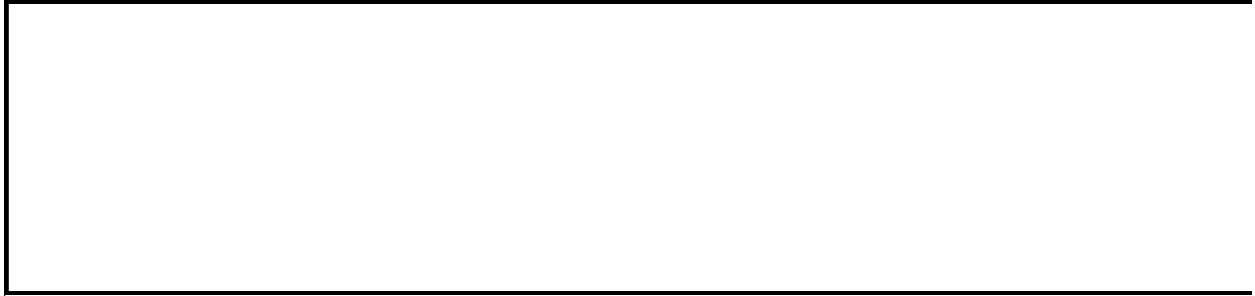
Copying begins at the current row of the **Recordset** object. After copying is completed, the **EOF** property of the **Recordset** object is **True**.

When this method copies the recordset to the worksheet, the results will be truncated if you do not specify a range that is large enough to hold the contents of the recordset.

## Example

This example copies a recordset named rstAuthors into the active sheet of Spreadsheet1 starting at cell A1.

```
Spreadsheet1.ActiveSheet.Cells.CopyFromRecordset rstAuthors
```



# Cut Method

-

Cuts the specified range and either moves it to the Clipboard or pastes it into a specified destination range.

*expression.Cut(Destination)*

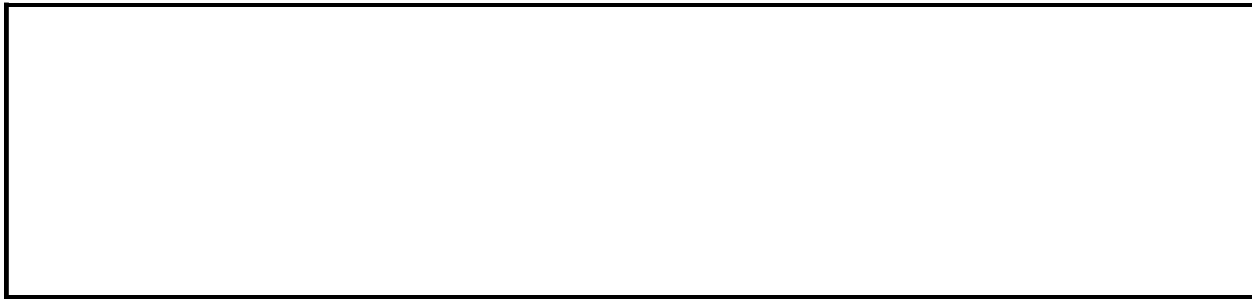
*expression* An expression that returns a [Range](#) object.

***Destination*** Optional **Variant**. If you do not specify this argument, the specified range is sent to the Clipboard. If this argument is a **Range** object, the source range is moved to the specified range.

## Example

This example moves the formula, data, and formatting from cell A3 to cell G7 on the active worksheet.

```
Sub MoveCell()  
    Dim shtSource  
  
    Set shtSource = Spreadsheet1.ActiveSheet  
  
    ' Move the contents of cell A3 to cell G7.  
    shtSource.Range("A3").Cut shtSource.Range("G7")  
End Sub
```



▼ [Show All](#)

# Delete Method

▸ [Delete](#) method as it applies to the [Hyperlink](#), [Name](#), [PivotHyperlink](#), [Sheets](#), [Worksheet](#), and [Worksheets](#) objects.

Deletes the specified object.

*expression*.**Delete**

*expression* An expression that returns one of the objects listed above.

▸ [Delete](#) method as it applies to the [ChAxes](#), [ChCharts](#), [ChSeriesCollection](#), [ElementExtensions](#), [GroupingDefs](#), [GroupLevels](#), [LookupRelationships](#), [PageFields](#), [ParameterValues](#), [RecordsetDefs](#), [SchemaFields](#), [SchemaRelationships](#), [SchemaRowsources](#), and [SublistRelationships](#) objects.

Deletes a single object from the specified collection.

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

*expression* An expression that returns one of the objects listed above.

**Index** Required **Variant**. Specifies the name or number of the object to be deleted.

▸ [Delete](#) method as it applies to the [ChDataLabelsCollection](#), [ChErrorBarsCollection](#), [ChSegments](#) and [ChTrendlines](#) objects.

Deletes a single object from the specified collection.

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

*expression* An expression that returns one of the objects listed above.

***Index*** Required **Long**. Specifies the name or number of the object to be deleted.

► [Delete method as it applies to the Range object.](#)

Deletes the specified cells from the worksheet.

*expression*.**Delete**(**Shift**)

*expression* An expression that returns one of the objects listed above.

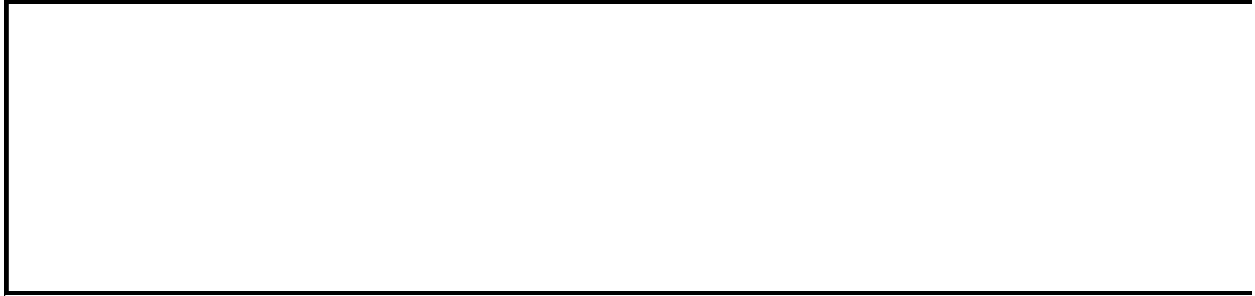
***Shift*** Optional **Variant**. Specifies how to shift cells to replace deleted cells. Can be one of the following **XlDeleteShiftDirection** constants: **xlShiftToLeft** or **xlShiftUp**. If this argument is omitted, Microsoft Excel decides based on the shape of the range.



## Example

This example deletes the data labels from the specified series in the chart workspace.

```
ChartSpace1.Charts(0).SeriesCollection(0).DataLabelsCollection.Delet
```



# DeleteCustomGroupMember Method

-  
Deletes a member from a custom group field.

*expression*.**DeleteCustomGroupMember**(*CustomGroupMember*)

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

***CustomGroupMember*** Required **Variant**. Name, unique name, or reference to the member to delete.

## Remarks

This method will return a run-time error if the member referred to by the *CustomGroupMember* argument is not a member of a custom group field.

# DeleteField Method

-

Deletes a calculated field that was created by the **AddCalculatedField** or **AddCustomGroupField** methods from the specified field set.

*expression*.**DeleteField**(*Field*)

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

***Field*** Required **Variant**. The name, unique name or reference to a calculated field in the field set.



# DeleteFieldSet Method

-

Deletes a field set that was created using the **AddFieldSet** method. The field set is removed from the PivotTable view and from the **PivotFieldSets** collection.

*expression*.**DeleteFieldSet**(*FieldSet*)

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

*FieldSet* Required **Variant**. The name of, or a reference to, a custom field set.

## Example

The following line of code deletes a custom field set named "Variance" from PivotTable1.

```
PivotTable1.ActiveView.DeleteFieldSet "Variance"
```

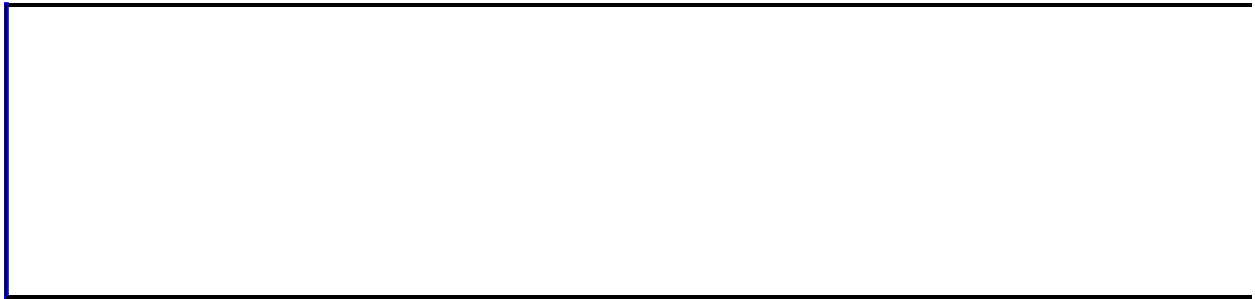
---

# DeleteRecord Method

-  
Deletes a record from the specified data page.

*expression*.**DeleteRecord**

*expression* An expression that returns a [DataPage](#) object.



# DeleteTotal Method

-  
Deletes a [PivotTotal](#) object from the [PivotTotals](#) collection. You can delete only user-defined totals.

*expression*.**DeleteTotal**(*Total*)

*expression* An expression that returns a [PivotView](#) object.

**Total** Required **Variant**. Specifies the name or number of the total.



## Example

This example deletes a total named "Total Budget" from PivotTable1.

```
Sub Delete_Total()  
    Dim vwView  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' Delete the total named "Total Budget."  
    vwView.DeleteTotal vwView.Totals("Total Budget")  
End Sub
```



# DrawEllipse Method

-

Draws an ellipse on the specified chart. Use the current settings of the **Border** and **Interior** properties to determine the properties of the new ellipse.

*expression*.**DrawEllipse**(*Left*, *Top*, *Right*, *Bottom*)

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

**Left** Required **Long**. Pixel coordinate of the left edge of the ellipse.

**Top** Required **Long**. Pixel coordinate of the top edge of the ellipse.

**Right** Required **Long**. Pixel coordinate of the right edge of the ellipse.

**Bottom** Required **Long**. Pixel coordinate of the bottom edge of the ellipse.

## Example

This example uses the BeforeRender event to cancel drawing the gridlines and plot area of the first chart in ChartSpace1. The AfterRender event then replaces the plot area with an ellipse that is drawn after the chart is rendered.

```
Private Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chConstants

    Set chConstants = ChartSpace1.Constants

    ' Check to see if the chart has been rendered.
    If TypeName(chartObject) = "ChChart" Then

        ' The next three lines of code set the interior
        ' and border properties of the ellipse.
        drawObject.Interior.SetPresetGradient _
            chConstants.chGradientHorizontal, _
            chConstants.chGradientVariantStart, _
            Int((24 - 1 + 1) * Rnd + 1)
        drawObject.Border.Weight = 1
        drawObject.Border.Color = "black"

        ' Begin the drawing object.
        drawObject.BeginObject 1

        ' Draw the ellipse.
        drawObject.DrawEllipse chartObject.Left, chartObject.Bottom,
                                chartObject.Right, chartObject.Top

        drawObject.EndObject

    End If

End Sub

Private Sub ChartSpace1_BeforeRender(chartObject, Cancel)

    Select Case TypeName(chartObject)

        Case "ChGridlines"

            ' Cancel the drawing of the gridlines.
            Cancel.Value = True

    End Select

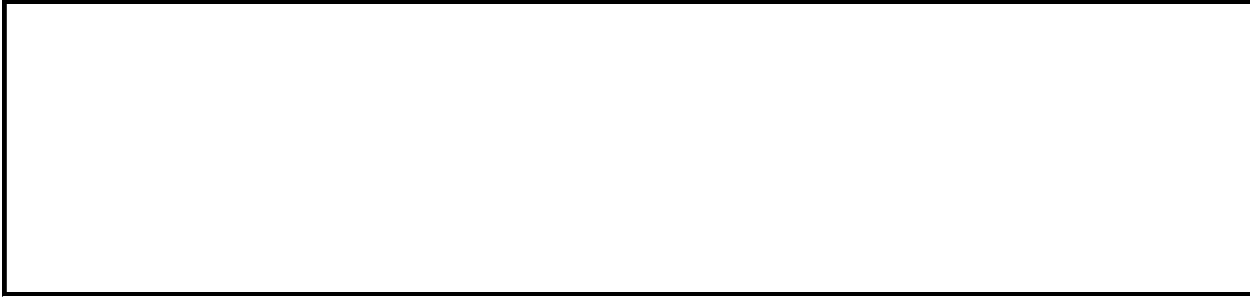
End Sub
```

```
Case "ChPlotArea"
```

```
    ' Cancel the drawing of the plot area.  
    Cancel.Value = True
```

```
End Select
```

```
End Sub
```



# DrawLine Method

-

Draws a line on the specified chart. Uses the current settings of the **Line** property to determine the properties of the new line.

*expression*.**DrawLine**(*x0*, *y0*, *x1*, *y1*)

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

*x0* Required **Long**. Starting pixel coordinate in the X plane.

*y0* Required **Long**. Starting pixel coordinate in the Y plane.

*x1* Required **Long**. Ending pixel coordinate in the X plane.

*y1* Required **Long**. Ending pixel coordinate in the Y plane.

## Example

This example illustrates how you can use the BeforeRender and AfterRender events together to create custom gridlines. The BeforeRender event cancels the rendering of the gridlines and the AfterRender event draws custom gridlines.

```
Sub ChartSpace1_BeforeRender(chartObject, Cancel)

    ' Check to see if the next object to be rendered
    ' is a gridline.
    If TypeName(chartObject) = "ChGridlines" Then

        ' Cancel the rendering of gridlines.
        Cancel.Value = True

    End If

End Sub

Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chChart1
    Dim plPlotArea
    Dim lLeft
    Dim lRight
    Dim lHeight
    Dim lTop
    Dim lIncrement
    Dim chConstants
    Dim iCtr

    Set chConstants = ChartSpace1.Constants

    ' Set a variable to the first chart in Chartspace1.
    Set chChart1 = ChartSpace1.Charts(0)

    ' Set a variable to the plot area of the chart.
    Set plPlotArea = chChart1.PlotArea

    ' Check to see if the rendered object is a gridline.
    If TypeName(chartObject) = "ChGridlines" Then

        ' The next four lines of code use the extents of
        ' the plot area to calculate the dimensions of the line.
        ' to be drawn.
        lLeft = plPlotArea.Left
```

```
lTop = plPlotArea.Top
lRight = plPlotArea.Right
lHeight = plPlotArea.Bottom - lTop

' Determine the increment to use between gridlines.
' Change the divisor to adjust the increment.
lIncrement = lHeight / 10

' The next three lines of code set the properties of the
' line to be drawn.
drawObject.Line.DashStyle = chConstants.chLineRoundDot
drawObject.Line.Color = "Green"
drawObject.Line.Weight = chConstants.owcLineWeightMedium

For iCtr = 1 To 9

    ' Draw the line.
    drawObject.DrawLine lLeft, lTop + iCtr * lIncrement, _
                        lRight, lTop + iCtr * lIncrement

Next

End If

End Sub
```



# DrawPolygon Method

-

Draws a polygon on the chart. The points for the polygon are specified in arrays containing the X and Y values that describe the segments of the polygon. Uses the current settings of the **Border** and **Interior** properties to determine the properties of the new polygon.

*expression*.**DrawPolygon**(*xValues*, *yValues*)

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

**xValues** Required **Variant**. An array containing the X values used to calculate the polygon.

**yValues** Required **Variant**. An array containing the Y values used to calculate the polygon.



This example uses the `BeforeRender` event to cancel rendering the chart title and the `AfterRender` event to replace the chart title with a polygon.

```
Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim alXValues(9)
    Dim alYValues(9)
    Dim chConstants
    Dim iCutoff

    iCutoff = 20

    Set chConstants = ChartSpace1.Constants

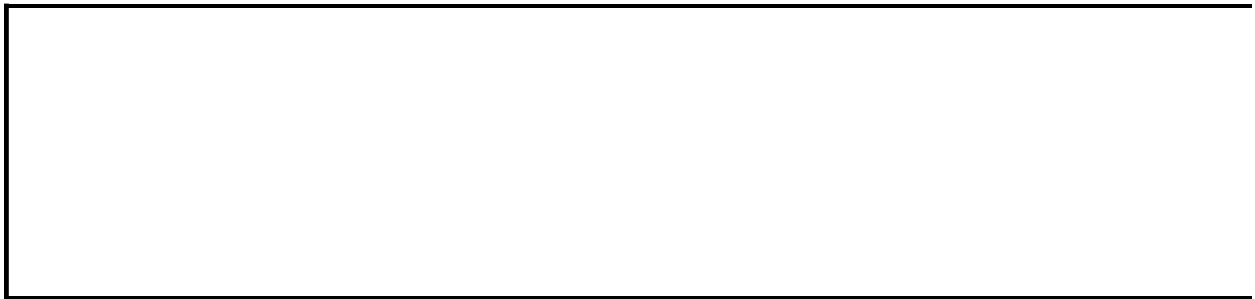
    If TypeName(chartObject) = "ChTitle" Then

        ' Set the array containing the x values for
        ' the line.
        alXValues(0) = chartObject.Left + iCutoff
        alXValues(1) = chartObject.Right - iCutoff
        alXValues(2) = chartObject.Right
        alXValues(3) = chartObject.Right
        alXValues(4) = chartObject.Right - iCutoff
        alXValues(5) = chartObject.Left + iCutoff
        alXValues(6) = chartObject.Left
        alXValues(7) = chartObject.Left
        alXValues(8) = chartObject.Left + iCutoff

        ' Set the array containing the y values for
        ' the line.
        alYValues(0) = chartObject.Top
        alYValues(1) = chartObject.Top
        alYValues(2) = chartObject.Top + iCutoff
        alYValues(3) = chartObject.Bottom - iCutoff
        alYValues(4) = chartObject.Bottom
        alYValues(5) = chartObject.Bottom
        alYValues(6) = chartObject.Bottom - iCutoff
        alYValues(7) = chartObject.Top + iCutoff
        alYValues(8) = chartObject.Top

        ' Set the properties for the polygon.
        drawObject.Interior.SetTwoColorGradient chConstants.chGradie
            chConstants.chGradientVariantStart
```

```
        ' Draw the polygon.  
        drawObject.DrawPolygon alXValues, alYValues  
  
    End If  
  
End Sub  
  
Private Sub ChartSpace1_BeforeRender(chartObject,Cancel)  
    If TypeName(chartObject) = "ChTitle" Then  
        Cancel.Value = True  
    End If  
  
End Sub
```



# DrawPolyLine Method

-

Draws a line containing multiple segments. The points for the line are specified in arrays containing the X and Y values that describe the segments of the line. Uses the current settings of the **Line** property to determine the properties of the new line.

*expression*.**DrawPolyLine**(*xValues*, *yValues*)

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

**xValues** Required **Variant**. An array containing the X values used to calculate the line.

**yValues** Required **Variant**. An array containing the Y values used to calculate the line.

## Example

This example uses the AfterRender event to draw a custom border around ChartSpace1.

```
Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim alXValues(9)
    Dim alYValues(9)
    Dim iCutOff
    Dim chConstants

    iCutOff = 10

    Set chConstants = ChartSpace1.Constants

    If TypeName(chartObject) = "ChChart" Then

        ' Set the array containing the x values for
        ' the line.
        alXValues(0) = chartObject.Left + iCutOff
        alXValues(1) = chartObject.Right - iCutOff
        alXValues(2) = chartObject.Right
        alXValues(3) = chartObject.Right
        alXValues(4) = chartObject.Right - iCutOff
        alXValues(5) = chartObject.Left + iCutOff
        alXValues(6) = chartObject.Left
        alXValues(7) = chartObject.Left
        alXValues(8) = chartObject.Left + iCutOff

        ' Set the array containing the y values for
        ' the line.
        alYValues(0) = chartObject.Top
        alYValues(1) = chartObject.Top
        alYValues(2) = chartObject.Top + iCutOff
        alYValues(3) = chartObject.Bottom - iCutOff
        alYValues(4) = chartObject.Bottom
        alYValues(5) = chartObject.Bottom
        alYValues(6) = chartObject.Bottom - iCutOff
        alYValues(7) = chartObject.Top + iCutOff
        alYValues(8) = chartObject.Top

        ' Set the properties for the line.
        drawObject.Line.Color = "blue"
        drawObject.Line.Weight = chConstants.owcLineWeightThick
        drawObject.Line.DashStyle = chConstants.chLineLongDashDot
```

```
' Draw the line.  
drawObject.DrawPolyLine alXValues, alYValues
```

```
End If
```

```
End Sub
```



# DrawRectangle Method

-

Draws a rectangle on a chart. Uses the current settings of the **Border** and **Interior** properties to determine the properties of the new rectangle.

*expression*.**DrawRectangle**(*Left*, *Top*, *Right*, *Bottom*)

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

**Left** Required **Long**. Pixel coordinate of the left edge of the rectangle.

**Top** Required **Long**. Pixel coordinate of the top edge of the rectangle.

**Right** Required **Long**. Pixel coordinate of the right edge of the rectangle.

**Bottom** Required **Long**. Pixel coordinate of the bottom edge of the rectangle.

## Example

This example uses the `AfterRender` event to draw rectangles as a substitute for the legend entries in the first chart of `Chartspace1`.

```
Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chConstants

    Set chConstants = ChartSpace1.Constants

    If TypeName(chartObject) = "ChLegendEntry" Then

        ' Set the interior of the rectangle to a preset texture.
        ' You could substitute a URL to a custom graphic
        ' for the texture.
        drawObject.Interior.SetTextured chConstants.chTextureSand

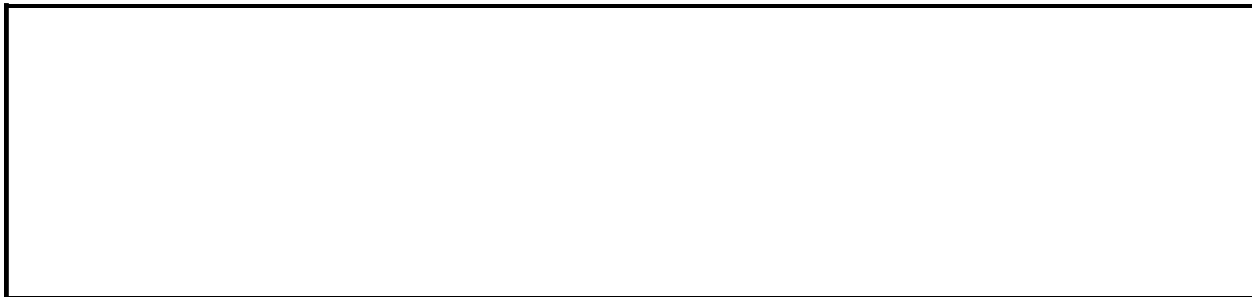
        ' Begin drawing the rectangle.
        drawObject.BeginObject 1

        ' Draw the rectangle.
        drawObject.DrawRectangle chartObject.Left, chartObject.Top,
                                chartObject.Right, chartObject.Bott

        drawObject.EndObject

    End If

End Sub
```



# DrawText Method

-  
Draws a text string on a chart.

*expression*.**DrawText**(*bstrText*, *Left*, *Top*)

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

*bstrText* Required **String**. The text to draw on the chart.

*Left* Required **Long**. Pixel coordinate of the left edge of the text.

*Top* Required **Long**. Pixel coordinate of the top edge of the text.



## Example

This example adds a text string to the upper-left corner of the plot area each time that the chart is re-drawn.

```
Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chChart1

    Set chChart1 = ChartSpace1.Charts(0)

    ' After the legend has been rendered, then add the text
    ' to the chart.
    If TypeName(chartObject) = "ChLegend" Then
        drawObject.DrawText "2000 Sales", chChart1.PlotArea.Left + 5
                                chChart1.PlotArea.Top
    End If

End Sub
```



▼ [Show All](#)

# DropZones Method

-

Returns a **ChDropZone** object. Use the properties of the returned object to format the drop zone.

*expression*.**DropZones**(*dz*)

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

*dz* Required [ChartDropZonesEnum](#). Represents the drop zone that you want to format.

ChartDropZonesEnum can be one of these ChartDropZonesEnum constants.

**chDropZoneCategories**

**chDropZoneCharts**

**chDropZoneData**

**chDropZoneFilter**

**chDropZoneSeries**

## Example

This example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
  
    Dim dzSeriesDropZone  
    Dim chConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```



# DuplicateFormat Method

-

Copies the formatting and type of the specified chart to other charts in the workspace that are bound to the same data source. This method can only be used when the chart control is bound to a relational data source list.

*expression*.**DuplicateFormat**

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

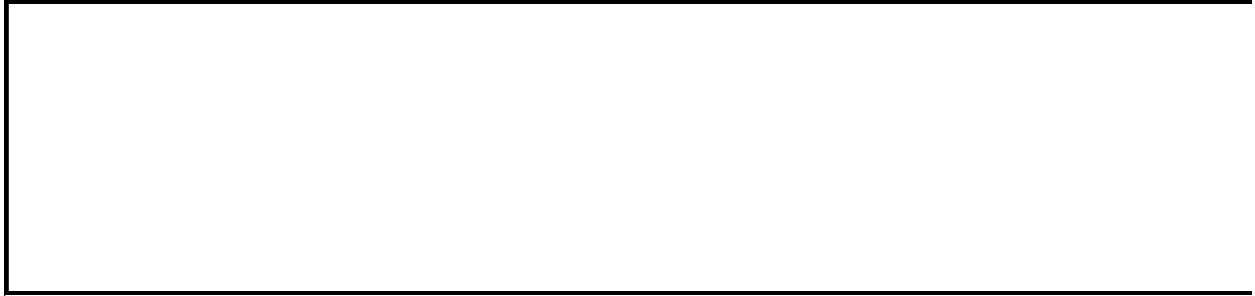
## Remarks

The **HasMultipleCharts** property of the chart control must be set to **True** before using this method.

## Example

This example copies the formatting of the first chart in Chartspace1 to all other charts in Chartspace1 that are bound to the same PivotTable list.

```
Chartspace1.Charts(0).DuplicateFormat
```



# EndEdit Method

-

Moves the active cell out of edit mode.

*expression*.**EndEdit**(*Accept*)

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

**Accept** Optional **Boolean**. Specifies whether the current value is accepted. Setting this argument to **False** discards any changes to the detail cell and the previous value is restored. The default value is **True**.





# EndObject Method

-  
Ends the drawing sequence for the specified **ChChartDraw** object.

*expression*.**EndObject**

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

## Example

This example uses the BeforeRender event to cancel drawing the gridlines and plot area of the first chart in ChartSpace1. The AfterRender event then replaces the plot area with an ellipse that is drawn after the chart is rendered.

```
Private Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chConstants

    Set chConstants = ChartSpace1.Constants

    ' Check to see if the chart has been rendered.
    If TypeName(chartObject) = "ChChart" Then

        ' The next three lines of code set the interior
        ' and border properties of the ellipse.
        drawObject.Interior.SetPresetGradient _
            chConstants.chGradientHorizontal, _
            chConstants.chGradientVariantStart, _
            Int((24 - 1 + 1) * Rnd + 1)
        drawObject.Border.Weight = 1
        drawObject.Border.Color = "black"

        ' Begin the drawing object.
        drawObject.BeginObject 1

        ' Draw the ellipse.
        drawObject.DrawEllipse chartObject.Left, chartObject.Bottom,
                               chartObject.Right, chartObject.Top

        drawObject.EndObject

    End If

End Sub

Private Sub ChartSpace1_BeforeRender(chartObject, Cancel)

    Select Case TypeName(chartObject)

        Case "ChGridlines"

            ' Cancel the drawing of the gridlines.
            Cancel.Value = True

    End Select

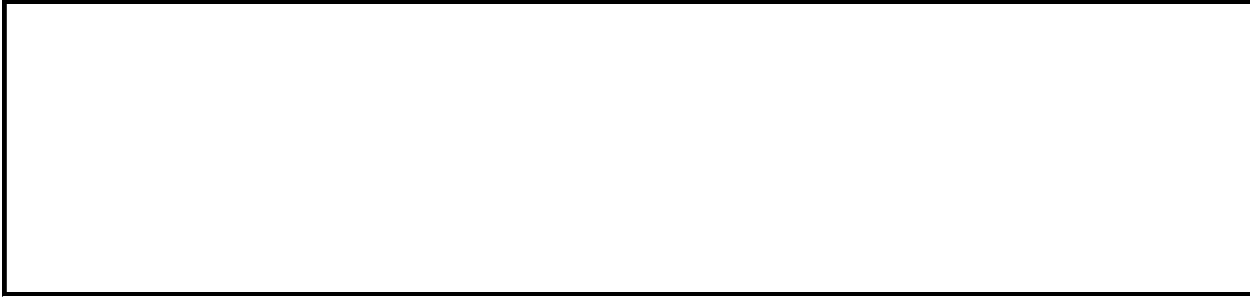
End Sub
```

```
Case "ChPlotArea"
```

```
    ' Cancel the drawing of the plot area.  
    Cancel.Value = True
```

```
End Select
```

```
End Sub
```



# EndUndo Method

-

Specifies the end of an undo block. This means that all statements between this call and its corresponding [BeginUndo](#) method call will be undone by a single call to the [Undo](#) method. This makes it possible for you to combine entire macros into one statement that can be easily undone. Undo blocks can be nested.

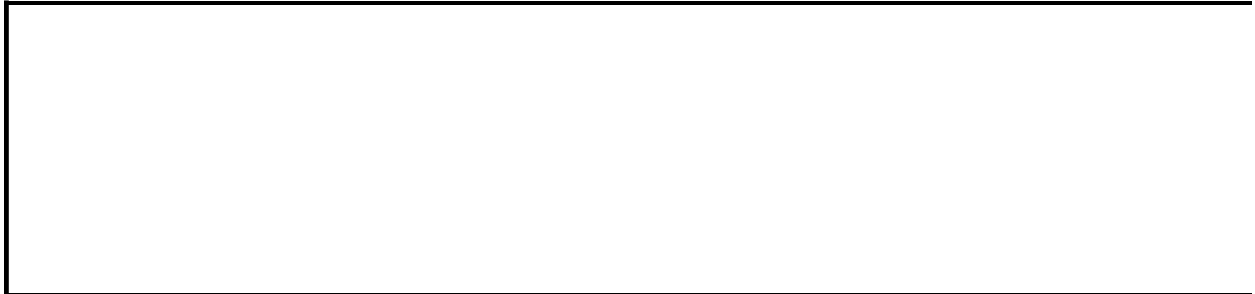
*expression*.**EndUndo**

*expression* An expression that returns a [ChartSpace](#) or [Spreadsheet](#) object.

## Example

This example creates an undo block containing code that sets the number format and font for cell D10. You can undo all of the formatting by clicking Undo on Spreadsheet1's toolbar.

```
Sub UndoBlock()  
    Dim rngCurrent  
  
    ' Enable undo.  
    Spreadsheet1.EnableUndo = True  
  
    ' Start an undo block.  
    Spreadsheet1.BeginUndo  
  
        Set rngCurrent = Spreadsheet1.Worksheets("sheet1").Range("D1  
  
        ' The following three lines of code apply  
        ' various formatting to cell D10.  
        rngCurrent.NumberFormat = "0.###"  
        rngCurrent.Font.Color = "Blue"  
        rngCurrent.Font.Name = "Times New Roman"  
  
    ' End the undo block.  
    Spreadsheet1.EndUndo  
End Sub
```



# EuroConvert Method

-

You can use the **EuroConvert** method to convert a number to the euro or from the euro to a participating currency. You can also use it to convert a number from one participating currency to another by using the euro as an intermediary (triangulation). The **EuroConvert** method uses fixed conversion rates established by the European Commission. Returns a **Double** value.

*expression.euro Convert(Number, SourceCurrency, TargetCurrency, FullPrecision, TriangulationPrecision)*

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

**Number** Required **Double**. The number you want to convert.

**SourceCurrency** Required **String**. A string expression, or reference to a field containing the string, corresponding to the International Standards Organization (ISO) acronym for the currency you want to convert. Can be one of the ISO codes listed in the following table.

Currency	ISO Code	Calculation Precision	Display Precision
Belgian franc	BEF	0	0
Luxembourg franc	LUF	0	0
Deutsche mark	DEM	2	2
Spanish peseta	ESP	0	0
French franc	FRF	2	2
Irish punt	IEP	2	2
Italian lira	ITL	0	0
Netherlands guilder	NLG	2	2
Austrian schilling	ATS	2	2
Portuguese escudo	PTE	1	2

finish Markka	FIM	2	2
euro	EUR	2	2

In the preceding table, the calculation precision determines what currency unit to round the result to based on the conversion currency. For example, when converting to Deutsche marks, the calculation precision is 2, and the result is rounded to the nearest pfennig, 100 pfennigs to a mark. The display precision determines how many decimal places appear in the field containing the result.

Later versions of the **Euro Convert** method may support additional currencies.

Currency	ISO Code
Danish Krone	DKK
Drachma	GRD
Swedish Krona	SEK
Pound Sterling	GBP

**TargetCurrency** Required **String**. A three-letter string corresponding to the ISO code of the currency to which you want to convert the number. See the previous table for the ISO codes. For a list of ISO codes, see the **SourceCurrency** argument description.

**FullPrecision** Optional **Variant**. A logical value (**True** or **False**), or an expression that evaluates to a value of **True** or **False**, that specifies how to display the result.

#### Use

#### If you want to

Display the result with the currency-specific rounding rules (see the table in the **SourceCurrency** argument description). The calculation precision value is used to calculate the result and the display precision value to display the result. **False** is the default if the **FullPrecision** argument is omitted.

**True** Display the result with all significant digits resulting from the calculation.

**TriangulationPrecision** Optional **Variant**. A value greater than or equal to 3 that specifies the number of significant digits in the calculation precision used for the intermediate euro value when converting between two national currencies.

## Remarks

Any trailing zeros are truncated and invalid parameters return #Error.

If the source ISO code is the same as the target ISO code, the original value of the number is active.

This method does not apply a format.

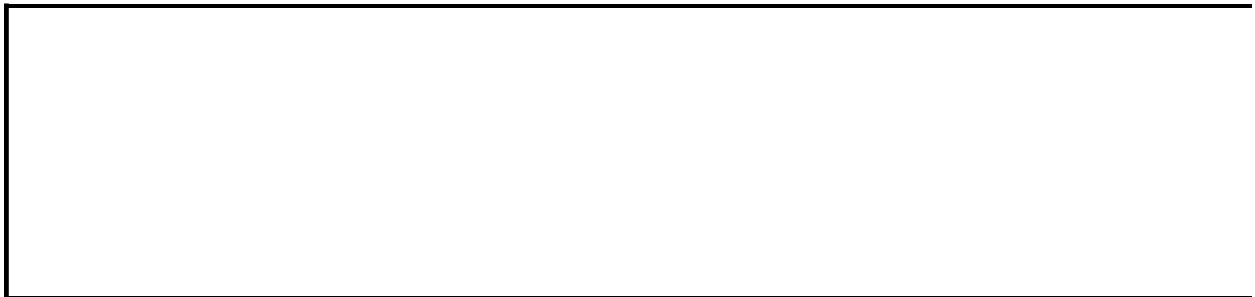
The **E uro Convert** method uses the current rates established by the European Commission. If the rates change, Microsoft will update the method. To get full information about the rules and the rates currently in effect, see the European Commission publications about the euro.



## Example

This example converts the value of the UnitPrice field from French francs to euros.

```
Dub ConvertToEuros()  
  
    Dim dblSourceNum  
    Dim dblConvertedNum  
  
    ' Set a variable to the UnitPrice field.  
    dblSourceNum = Document.All("unitprice").Value  
  
    ' Convert the UnitPrice from French francs to euros.  
    dblConvertedNum = MSODSC.EuroConvert(dblsourcenum, "FRF", "EUR", Fal  
  
    ' Place the converted value in the EuroValue field.  
    Document.All("EuroValue").Value = dblConvertedNum  
  
End Sub
```



# Evaluate Method

-

Evaluates an expression that is in the form of text and returns the result. The expression can include any combination of functions, keywords, or other syntax that the Spreadsheet Component can resolve.

*expression*.**Evaluate**(*Expression*)

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

***Expression*** Required **Variant**. The expression to evaluate.

## 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.

## Example

This example uses the the **Evaluate** method to calculate the cotangent of an angle.

```
Function CalcCotangent(sngAngleInDegrees)

    Dim strExpression

    ' Put together the expression to calculate the cotangent
    ' of the angle.
    strExpression = "1/TAN(" & sngAngleInDegrees & "*PI()/180)"

    ' Evaluate the string and return the result.
    CalcCotangent = Spreadsheet1.ActiveSheet.Evaluate(strExpression)

End Function
```



▼ [Show All](#)

# Execute Method

► [Execute method as it applies to the \*\*DataSourceControl\*\* object.](#)

Returns a **Recordset** object that represents the recordset resulting from a recordset definition or grouping definition. If the recordset is part of a hierarchy of chaptered recordsets, the data source control creates the entire hierarchy and returns the requested recordset from the hierarchy.

*expression*.**Execute**(*RecordsetName*, *ExecuteOption*, *FetchType*)

*expression* An expression that returns a **DataSourceControl** object.

**RecordsetName** Required **String**. The name of the recordset definition or grouping definition.

**ExecuteOption** Optional ADO **ExecuteOptionEnum**. Specifies how the query is performed. The default value is **adOptionUnspecified**.


**FetchType** Optional **dscFetchTypeEnum**. If the specified recordset definition contains sublist child recordset definitions, setting this argument to **dscFull** causes all records to be fetched whenever the **Execute** method is called. If you do not specify this argument or you set it to **dscParameterized**, records in child recordsets are fetched only when the child recordset is opened for a given parent. For example, if you have a Customers parent with an Orders child and this argument is set to **dscFull**, all records for all customers are fetched at run time. If this argument is set to **dscParameterized**, recordsets for customers with Customer ID 'ANTAR' are fetched only when the Customers recordset is positioned on this Customer ID and the child Orders recordset is opened.

► [Execute method as it applies to the \*\*OCCommand\*\* object.](#)

Executes the specified command.

*expression*.**Execute**( )

*expression* An expression that returns an **OCCommand** object.



# Expand Method

-

Causes the specified section to expand.

*expression*.**Expand**

*expression* An expression that returns a [Section](#) object.





▼ [Show All](#)

# Export Method

▸ [Export method as it applies to the \*\*PivotTable\*\* object.](#)

Saves the specified PivotTable list to a file and optionally opens it in Microsoft Excel.

*expression*.**Export**(*Filename*, *Action*)

*expression* An expression that returns a **PivotTable** object.

**Filename** Optional **String**. Specifies the file name of the saved file. If you do not specify this argument, a temporary file is created in the user's temporary folder (the location of the temporary folder varies by operating system).

**Action** [PivotExportActionEnum](#). Specifies what happens after the PivotTable list is saved to a file. If you do not specify this argument, the PivotTable list is opened in Excel. If Excel is not installed on the user's machine, an alert is displayed.

PivotExportActionEnum can be one of these PivotExportActionEnum constants.

**plExportActionNone**

**plExportActionOpenInExcel** *default*

▸ [Export method as it applies to the \*\*Spreadsheet\*\* object.](#)

*expression*.**Export**(*Filename*, *Action*, *Format*)

*expression* An expression that returns a **Spreadsheet** object.

**Filename** Optional **String**. Specifies the file name of the saved file. If you do

not specify this argument, a temporary file is created in the user's temporary folder (the location of the temporary folder varies by operating system). You must specify this argument if the **Action** argument is set to **ssExportActionNone**.

**Action** Optional [SheetExportActionEnum](#). Specifies whether or not the worksheet is saved to a file. If you do not specify this argument, the worksheet is opened in Microsoft Excel. If Excel is not installed on the user's machine, an alert is displayed.

SheetExportActionEnum can be one of these SheetExportActionEnum constants.

**ssExportActionNone**

**ssExportActionOpenInExcel** *default*

**Format** Optional [SheetExportFormat](#). Specifies the format to use when exporting the spreadsheet.

SheetExportActionEnum can be one of these SheetExportActionEnum constants.

**ssExportAsAppropriate** *default*

**ssExportHTML**

**ssExportXMLSpreadsheet**

## Example

► [As it applies to the \*\*Spreadsheet\*\* object.](#)

This example exports the active worksheet to the specified file.

```
Sub ExportSpreadsheet()  
    Dim ssConstants  
    Set ssConstants = Spreadsheet1.Constants  
    Spreadsheet1.Export "ssheet1.htm", ssConstants.ssExportActionNon  
        ssConstants.ssExportXMLSpreadsheet  
End Sub
```



# ExportPicture Method

-

Saves the specified chart workspace or PivotTable list as a graphics file.

*expression.ExportPicture(FileName, FilterName, Width, Height)*

*expression* An expression that returns a [ChartSpace](#) or [PivotTable](#) object.

**FileName** Optional **String**. Specifies the name of the saved file. If you do not specify this argument, the default file name is “Chart.gif” for a chart workspace or “Pivot.gif” for a PivotTable list.

**FilterName** Optional **String**. Specifies the name of the graphics filter that is used. Supported filter names are GIF, JPG, and PNG. The default is GIF.

**Width** Optional **Long**. Specifies the width of the graphic, in pixels. You must specify this argument for server-side charts.

**Height** Optional **Long**. Specifies the height of the graphic, in pixels. You must specify this argument for server-side charts.

## Example

This example saves the chart workspace as a graphics file.

```
ChartSpace1.ExportPicture "sales.gif", width:=320, height:=240
```



▼ [Show All](#)

# ExportXML Method

-  
Saves the current recordset as an XML file.

*expression.ExportXML(XMLDataTransformFile, eEncoding)*

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

*XMLDataTransformFile* Optional **Variant**. A path to an XSLT file.

*eEncoding* Optional [DscEncodingEnum](#). The format in which schema information is exported. If this argument is omitted, schema information is embedded in the data document.

DscEncodingEnum can be one of these DscEncodingEnum constants.

**dscEUCJ**

**dscUCS2**

**dscUCS4**

**dscUTF16**

**dscUTF8** *default*

**dscWindows**



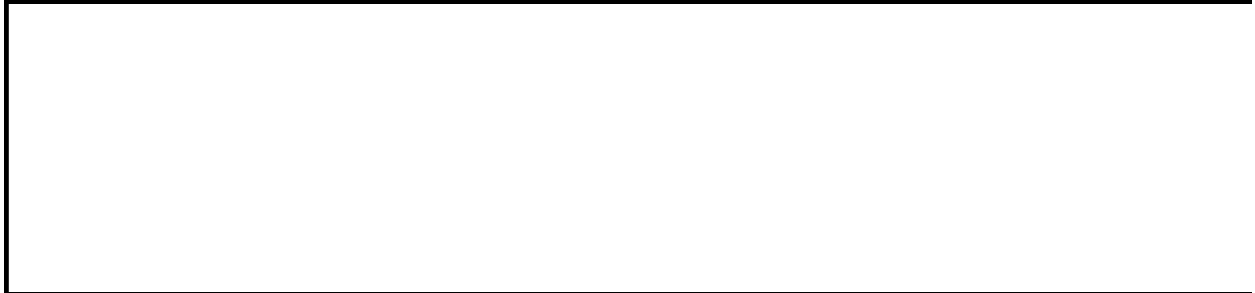
## Remarks

Use the **XMLLocation** property to set whether the data is exported to an XML data island or a separate XML data file. Use the **XMLDataTarget** property to specify the path or ID to use when exporting the data.

## Example

This example exports the current data in the data source control named MSODSC to an XML data file.

```
Sub ExportData()  
    Dim dscConstants  
  
    Set dscConstants = MSODSC.Constants  
  
    ' Set the location of the XML data to a data file.  
    MSODSC.XMLLocation = dscConstants.dscXMLDataFile  
  
    ' Set the specific target to export to.  
    MSODSC.XMLDataTarget = "Q1 Sales Analysis.xml"  
  
    ' Export the current data.  
    MSODSC.ExportXML  
  
End Sub
```



# FillDown Method

-

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 are copied into the rest of the rows in the range.

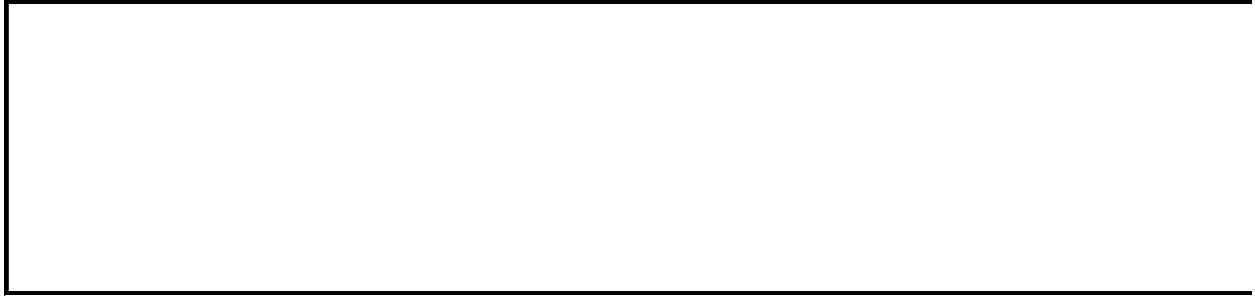
*expression*.**FillDown**

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

## Example

This example fills the range A1:A10 on the active worksheet, based on the contents of cell A1.

```
Spreadsheet1.ActiveSheet.Range("A1:A10").FillDown
```



# FillRight Method

-

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 are copied into the rest of the columns in the range.

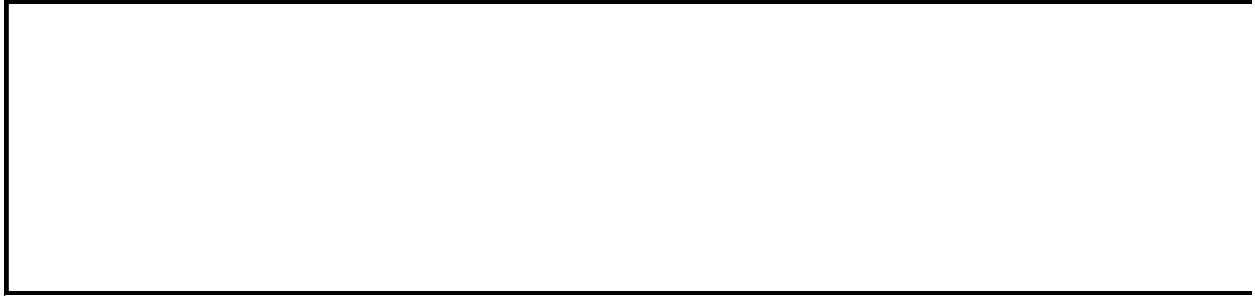
*expression*.**FillRight**

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

## Example

This example fills the range A1:M1 on the active worksheet, based on the contents of cell A1.

```
Spreadsheet1.ActiveSheet.Range("A1:M1").FillRight
```



▼ [Show All](#)

# Find Method

-

Finds specific information in a given range and returns a **Range** object that represents the first cell where that information is found. Returns **Nothing** if no match is found. Does not affect either the selection or the active cell.

*expression*.**Find**(*What*, *After*, *FindLookIn*, *FindLookAt*, *SearchOrder*, *SearchDirection*, *MatchCase*, *MatchByte*)

*expression* An expression that returns a [Range](#) object.

**What** Required **Variant**. The data to be searched for.

**After** Optional **Variant**. Specifies a single cell after which the search begins. This corresponds to the position of the active cell when a search is done from the user interface. Remember that the search begins *after* this cell; the specified cell isn't searched until the method wraps back around to this cell. If you don't specify this argument, the search starts after the cell in the upper-left corner of the range

**FindLookIn** Optional [XlFindLookIn](#). Specifies whether to search formulas or the displayed value.

XlFindLookIn can be one of these XlFindLookIn constants.

**xlFormulas**

**xlValues**

**FindLookAt** Optional [XlFindLookAt](#). Set this argument to **xlWhole** to force the entire contents of the cell to match the contents of the **What** argument.

XlFindLookAt can be one of these XlFindLookAt constants.

**xlPart**

**xlWhole**



**SearchOrder** Optional [XLSearchOrder](#). Specifies whether to search by columns or rows.

XLSearchOrder can be one of these XLSearchOrder constants.

**xlByColumns**

**xlByRows**

**SearchDirection** Optional [XLSearchDirection](#). Specifies the search direction.

XLSearchDirection can be one of these XLSearchDirection constants.

**xlNext**

**xlPrevious**

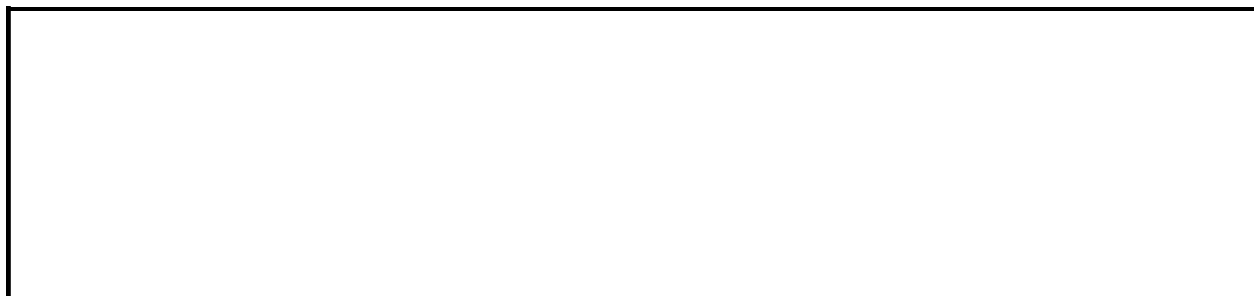
**MatchCase** Optional **Boolean**. **True** to make the search case sensitive. The default value is **False**.

**MatchByte** Optional **Variant**. Used only if you've selected or installed double-byte language support. **True** to have double-byte characters match only double-byte characters. **False** to have double-byte characters match their single-byte equivalents.

## Example

This example finds all occurrences of "Mike" in the range A1:F10 and makes those cells bold.

```
Sub Find_Mike()  
    Dim ssConstants  
    Dim rngFindRange  
    Dim rngFoundCell  
    Dim rngFirstFound  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Set a variable to the range to search.  
    Set rngFindRange = Spreadsheet1.Sheets("Sheet1").Range("A1:F10")  
  
    ' Find the first occurrence of Mike.  
    Set rngFoundCell = rngFindRange.Find("Mike", rngFindRange.Cells(  
        ssConstants.xlValues, ssConstants.xlPart)  
    )  
  
    ' If Mike was found...  
    If Not rngFoundCell Is Nothing Then  
  
        ' Set a variable to the first found instance.  
        Set rngFirstFound = rngFoundCell  
  
        Do  
            ' Set the font to bold.  
            rngFoundCell.Font.Bold = True  
  
            'Find the next occurrence of Mike.  
            Set rngFoundCell = rngFindRange.FindNext(rngFoundCell)  
  
            ' Loop until you return to the first occurrence of Mike.  
            Loop Until rngFoundCell.Address = rngFirstFound.Address  
        End If  
    End Sub
```



# FindNext Method

-

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.

*expression*.**FindNext**(*After*)

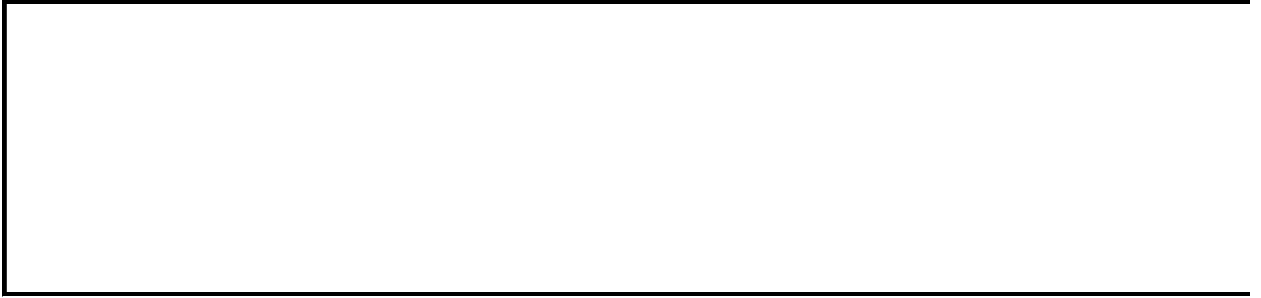
*expression* Required. An expression that returns a **Range** object.

**After** Optional **Variant**. The cell after which you want to search. This corresponds to the position of the active cell in the user interface. Note that **After** must be a single cell in the range. Remember that the search begins *after* the active cell; the active cell itself isn't searched until the **FindNext** method wraps back around to the active cell. If this argument isn't specified, the search starts after the cell in the upper-left corner of the range.

## Example

This example finds all occurrences of "Mike" in Sheet1 and makes those cells bold.

```
Sub FindMike()  
  
    Dim ssConstants  
    Dim rngFindRange  
    Dim rngFoundCell  
    Dim rngFirstFound  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Set a variable to the range to search.  
    Set rngFindRange = Spreadsheet1.Sheets("Sheet1").UsedRange  
  
    ' Find the first ocurence of Mike.  
    Set rngFoundCell = rngFindRange.Find("Mike", rngFindRange.Cells(  
        ssConstants.xlValues, ssConstants.xlPart)  
    )  
  
    ' If Mike was found...  
    If Not rngFoundCell Is Nothing Then  
  
        ' Set a variable to the first found instance.  
        Set rngFirstFound = rngFoundCell  
  
        Do  
            ' Set the font to bold.  
            rngFoundCell.Font.Bold = True  
  
            'Find the next occurrence of Mike.  
            Set rngFoundCell = rngFindRange.FindNext(rngFoundCell)  
  
            ' Loop until you return to the first occurrence of Mike.  
            Loop Until rngFoundCell.Address = rngFirstFound.Address  
  
        End If  
  
    End Sub
```



# FindPrevious Method

-

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.

*expression*.**FindPrevious**(*After*)

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

**After** Optional **Variant**. The cell before which you want to search. This corresponds to the position of the active cell in the user interface. Note that **After** must be a single cell in the range. Remember that the search begins *before* the active cell; the active cell itself isn't searched until the **FindPrevious** method wraps back around to this cell. If this argument isn't specified, the search starts before the upper-left cell in the range.

## 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 “Redmond” in column B.

```
Sub Find_Methods()  
  
    Dim rngFoundCell  
    Dim rngFindRange  
  
    ' Set a variable to the range to search.  
    Set rngFindRange = Spreadsheet1.ActiveSheet.Columns("B")  
  
    ' Find the first occurrence of Redmond in column B.  
    Set rngFoundCell = rngFindRange.Find("Redmond")  
  
    ' Display the location of the first occurrence of Redmond.  
    MsgBox "The first occurrence is in cell " & rngFoundCell.Address  
  
    ' Find the next occurrence of Redmond in column B.  
    Set rngFoundCell = rngFindRange.FindNext(after:=rngFoundCell)  
  
    ' Display the location of the next occurrence of Redmond.  
    MsgBox "The next occurrence is in cell " & rngFoundCell.Address  
  
    ' Find the previous occurrence of Redmond in column B.  
    Set rngFoundCell = rngFindRange.FindPrevious(after:=rngFoundCell)  
  
    ' Display the location of the previous occurrence of Redmond.  
    MsgBox "The previous occurrence is in cell " & rngFoundCell.Addr  
  
End Sub
```

---

# FlipHorizontal Method

-  
Flips all of the series in the specified chart horizontally.

*expression*.**FlipHorizontal**

*expression* An expression that returns a **ChPlotArea** object.



## Example

This example flips all of the series in the specified chart horizontally.

```
ChartSpace1.Charts(0).PlotArea.FlipHorizontal
```



# FlipVertical Method

-  
Flips all of the series in the specified chart vertically.

*expression*.**FlipVertical**

*expression* An expression that returns a **ChPlotArea** object.

## Example

This example flips all of the series in the specified chart vertically.

```
ChartSpace1.Charts(0).PlotArea.FlipVertical
```



▼ [Show All](#)

# Follow Method

► [Follow method as it applies to the \*\*Hyperlink\*\* object.](#)

Displays a cached document if it has already been downloaded. Otherwise, this method resolves the hyperlink, downloads the target document, and then displays the document.

*expression.Follow(NewWindow, AddHistory, ExtraInfo, Method, HeaderInfo)*

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

**NewWindow** Optional **Variant**. **True** to display the target document in a new window. The default value is **False**.

**AddHistory** Optional **Variant**. This argument is not supported.

**ExtraInfo** Optional **Variant**. This argument is not supported.

**Method** Optional **Variant**. This argument is not supported.

**HeaderInfo** Optional **Variant**. This argument is not supported.

► [Follow method as it applies to the \*\*PivotHyperlink\*\* object.](#)

Displays a cached document if it has already been downloaded. Otherwise, this method resolves the hyperlink, downloads the target document, and then displays the document.

*expression.Follow(NewWindow)*

*expression* An expression that returns a **PivotHyperlink** object.

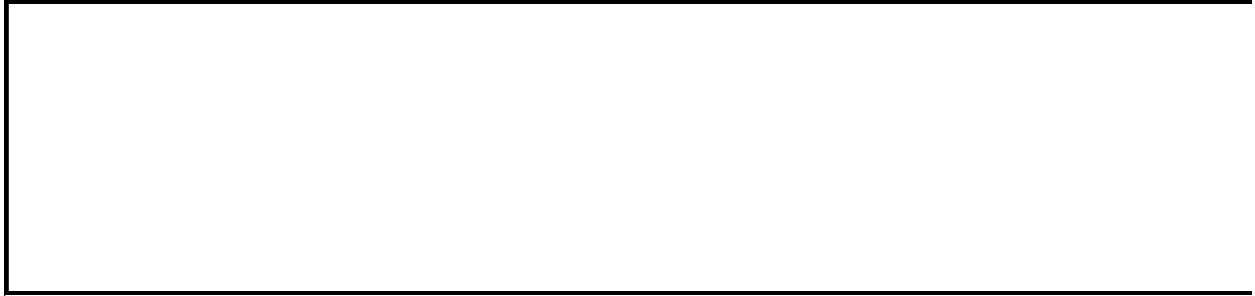
**NewWindow** Optional **Boolean**. **True** to display the target document in a new

window. The default value is **False**.

## Example

This example resolves the hyperlink in cell B15 on the active worksheet, downloads the target document, and then displays the document.

```
Spreadsheet1.ActiveSheet.Range("b15").Hyperlink.Follow
```



# GetContainingSection Method

-  
Returns the containing section for the specified HTML element.

*expression*.**GetContainingSection**(*Element*)

*expression* An expression that returns a [DataSourceControl](#) object.

**Element** Required **Object**. Represents the specified HTML element.





# GetPicture Method

-  
Returns a picture of a chart from a binary data stream.

*expression*.**GetPicture**(*FilterName*, *Width*, *Height*)

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

***FilterName*** Optional **String**. The name of the graphics filter to use. The default value is "GIF".

***Width*** Optional **Long**. The width of the chart in pixels.

***Height*** Optional **Long**. The height of the chart in pixels.

## Remarks

You can use the **BinaryWrite** method to write the picture returned by this method to the current HTTP session.

## Example

This example uses an ASP script to create a chart based on data in a SQL Server database. Once the chart has been created, an picture of the chart is displayed in the browser window.

```
<%
Dim PictType
Dim NewChart
Dim chConstants

Set NewChart = CreateObject("OWC10.ChartSpace")

Response.Expires = 0
Response.Buffer = True
Response.Clear

PictType = "jpg"
Response.ContentType = "image/" & PictType

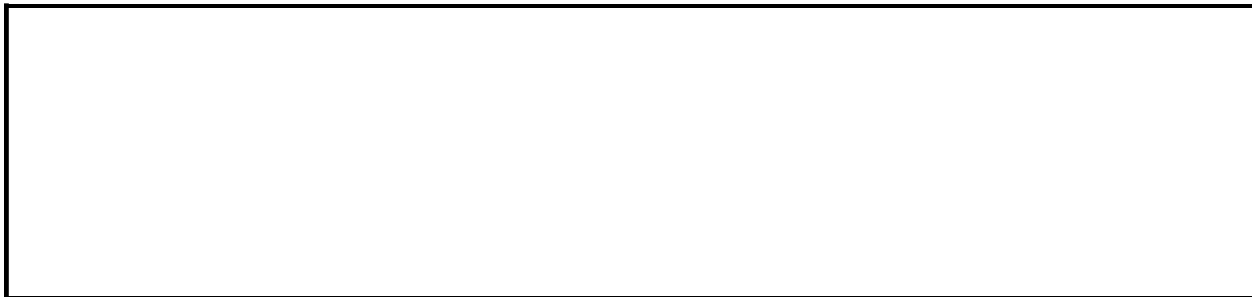
Set chConstants = NewChart.Constants

NewChart.ConnectionString = "Provider=SQLOLEDB.1;persist Security
                             "Catalog=Northwind;Data Source=server
NewChart.DataMember = "Order Details"

NewChart.SetData chConstants.chDimCategories, chConstants.chDataB
NewChart.SetData chConstants.chDimValues, chConstants.chDataBound

NewChart.Charts(0).Type = chConstants.chChartTypeColumn3D
NewChart.Charts(0).HasTitle = True
NewChart.Charts(0).Title.Caption = "Server-Rendered Chart"

Response.BinaryWrite NewChart.GetPicture(PictType, 500, 400)
%>
```



▼ [Show All](#)

# GetValue Method

Returns a data value for the specified point in a custom data dimension.

*expression*.**GetValue**(*Dimension*, *scaled*)

*expression* An expression that returns a **ChPoint** object.

**Dimension** Required [ChartDimensionsEnum](#). The dimension from which you are retrieving the value.

ChartDimensionsEnum can be one of these ChartDimensionsEnum constants.

**chDimBubbleValues**

**chDimCategories**

**chDimCharts**

**chDimCloseValues**

**chDimFilter**

**chDimFormatValues**

**chDimHighValues**

**chDimLowValues**

**chDimOpenValues**

**chDimRValues**

**chDimSeriesNames**

**chDimThetaValues**

**chDimValues**

**chDimXValues**

**chDimYValues**

**Scaled** Optional **Variant**. Specifies whether to return the actual value of the point, or its percentage as it relates to the other related points. Set this argument to **True** to return the percentage. This argument is relevant only when you are using a Pie, Doughnut, or Stacked Column chart.

## Remarks

Alternatively, you can specify a [ChartErrorBarCustomValuesEnum](#) constant for the ***Dimension*** argument to return the value of an error bar.

ChartErrorBarCustomValuesEnum can be one of these  
ChartErrorBarCustomValuesEnum constants.

**chErrorBarMinusValues chErrorBarPlusValues**

---

--

# Group Method

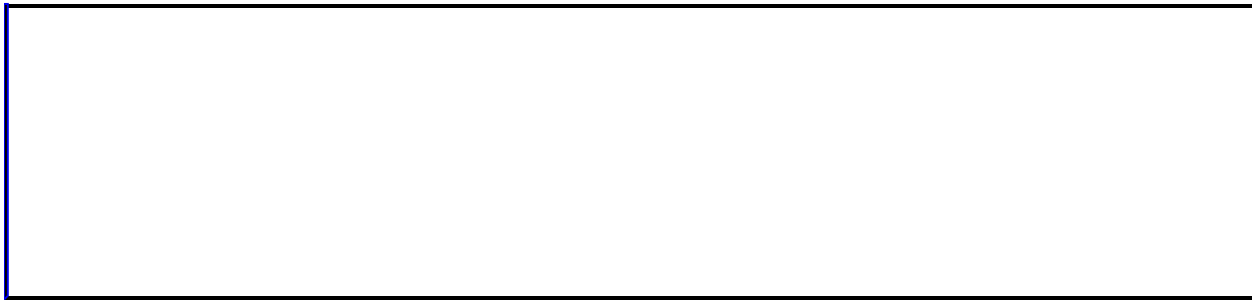
-

Groups the specified series with the series specified in the ***Series*** argument.

*expression*.**Group**(***Series***)

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

***Series*** Required **ChSeries** object. The series to group the specified series with.



# HideDetails Method

-

Hides the details cells for the specified object. If the specified object is a **PivotData** object, all detail cells are hidden. If the specified object is a **PivotRowMember** object, then all details cells in that row are hidden. If the specified object is a **PivotColumnMember** object, then all details cells in that column are hidden.

*expression*.**HideDetails**

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



## Example

This example hides all of the detail cells in PivotTable1.

```
PivotTable1.ActiveData.HideDetails
```



# Insert Method

-

Inserts a cell or a range of cells into the worksheet and shifts other cells away to make space.

*expression.Insert(Shift)*

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



# InsertFieldSet Method

-

Inserts a field set on the specified axis.

*expression*.**InsertFieldSet**(*FieldSet*, *Before*, *Remove*)

*expression* An expression that returns a [PivotAxis](#), [PivotDataAxis](#), [PivotFilterAxis](#), or [PivotGroupAxis](#) object.

**FieldSet** Required [PivotFieldSet](#) object. Specifies the field set to be inserted.

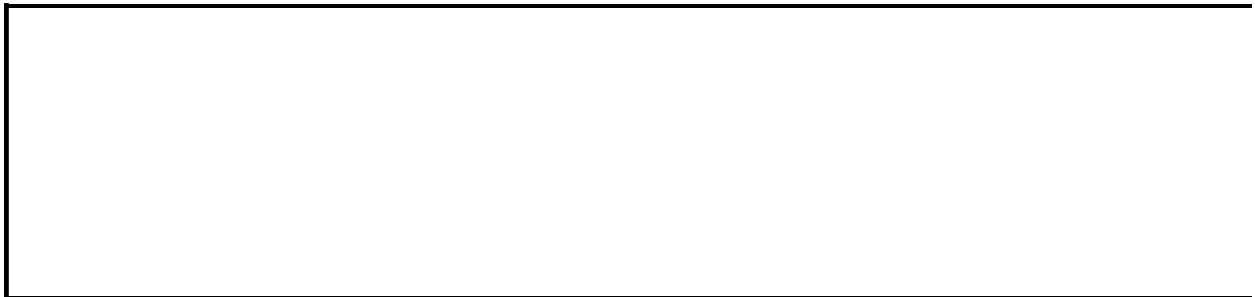
**Before** Optional **Variant**. Specifies the index of the field set before which the inserted field set will be placed.

**Remove** Optional **Boolean**. This argument is reserved for future use, and its value is always **True**. When the field set is added to the specified axis, it is removed from any other axis.

## Example

This example adds a fieldset to the row axis, data axis, and filter axis of PivotTable1.

```
Sub Add_Fields_To_PivotTable()  
    Dim vwView As PivotView  
    Dim ptConstants  
    Dim totOrderCount  
  
    Set ptConstants = PivotTable1.Constants  
    Set vwView = PivotTable1.ActiveView  
  
    ' Add the ShipCountry field to the row axis.  
    vwView.RowAxis.InsertFieldSet vwView.FieldSets("ShipCountry")  
  
    ' Add the OrderId field to the data axis.  
    vwView.DataAxis.InsertFieldSet vwView.FieldSets("OrderID")  
  
    ' Add the ShipVia field to the filter axis  
    vwView.FilterAxis.InsertFieldSet vwView.FieldSets("ShipVia")  
  
    ' Create a total named "Order Count" that counts the OrderID fie  
    Set totOrderCount = vwView.AddTotal("Order Count", vwView.Fields  
        ptConstants.plFunctionCount)  
  
    ' Add the Order Count total to the data axis.  
    vwView.DataAxis.InsertTotal totOrderCount  
End Sub
```



# InsertTotal Method

-  
Adds a [PivotTotal](#) object to the [PivotTotals](#) collection.

*expression*.**InsertTotal**(*Total*, *Before*)

*expression* An expression that returns a [PivotDataAxis](#) object.

**Total** Required [PivotTotal](#) object. Specifies the total to be inserted.

**Before** Optional **Variant**. Specifies the index of the total before which the inserted total will be placed. If you do not specify this argument, the total is inserted at the end of the collection.

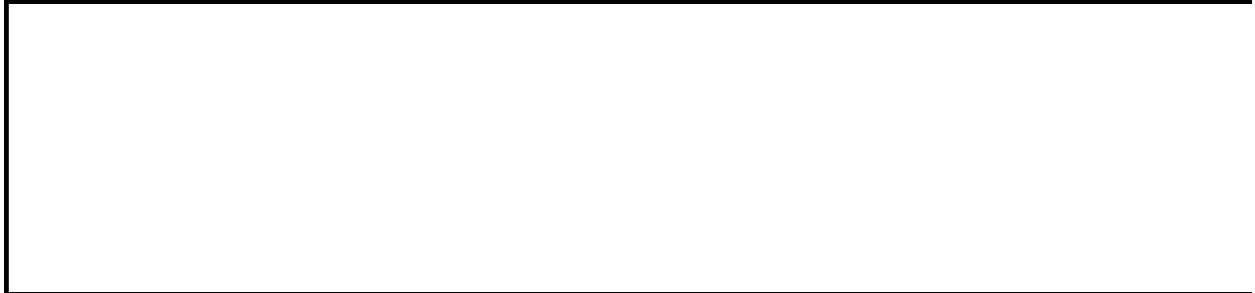
## Remarks

If the **PivotTotal** object is currently part of the **PivotTotals** collection, the object is first removed from that collection and then reinserted into it. This changes the display order because totals are displayed in their collection order.

## Example

This example adds a total named "Total Budget" that sums the values in the Budget field to PivotTable1, then then inserts the total into the PivotTable view.

```
Sub Add_Total()  
    Dim vwView  
    Dim ptConstants  
    Dim totNewTotal  
  
    Set vwView = PivotTable1.ActiveView  
    Set ptConstants = PivotTable1.Constants  
  
    ' Add a new total named "Total Budget" to the current view.  
    Set totNewTotal = vwView.AddTotal("Total Budget", vwView.Fields(ptConstants.plFunctionSum))  
  
    ' Insert the newly created total into the detail area of the  
    vwView.DataAxis.InsertTotal totNewTotal  
  
End Sub
```



▼ [Show All](#)



# IsButtonEnabled Method

▸ [IsButtonEnabled method as it applies to the \*\*DataPage\*\* object.](#)

Returns **True** if the specified navigation button is enabled.

*expression*.**IsButtonEnabled**(*Button*)

*expression* Required. An expression that returns a [DataPage](#) object.

*Button* Required [NavButtonEnum](#). Specifies the navigation button.

NavButtonEnum can be one of these NavButtonEnum constants.

**navbtnApplyFilter**

**navbtnDelete**

**navbtnHelp**

**navbtnMoveFirst**

**navbtnMoveLast**

**navbtnMoveNext**

**navbtnMovePrev**

**navbtnNew**

**navbtnSave**

**navbtnSaveAll**

**navbtnSortAscending**

**navbtnSortDescending**

**navbtnToggleFilter**

**navbtnUndo**

**navbtnUndoAll**

▸ [IsButtonEnabled method as it applies to the \*\*RecordNavigationControl\*\* object.](#)

Returns **True** if the specified navigation button is enabled.

*expression*.**IsButtonEnabled**(*navbtn*)

*expression* Required. An expression that returns a [RecordNavigationControl](#) object.

*navbtn* Required [NavButtonEnum](#). Specifies the navigation button.

NavButtonEnum can be one of these NavButtonEnum constants.

**navbtnApplyFilter**

**navbtnDelete**

**navbtnHelp**

**navbtnMoveFirst**

**navbtnMoveLast**

**navbtnMoveNext**

**navbtnMovePrev**

**navbtnNew**

**navbtnSave**

**navbtnSaveAll**

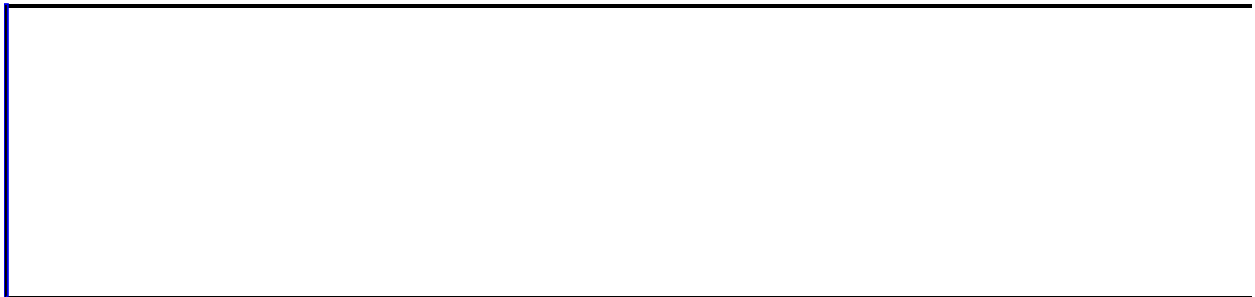
**navbtnSortAscending**

**navbtnSortDescending**

**navbtnToggleFilter**

**navbtnUndo**

**navbtnUndoAll**



# Item Method

-  
Returns a **Name** object from the **Names** collection.

*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.

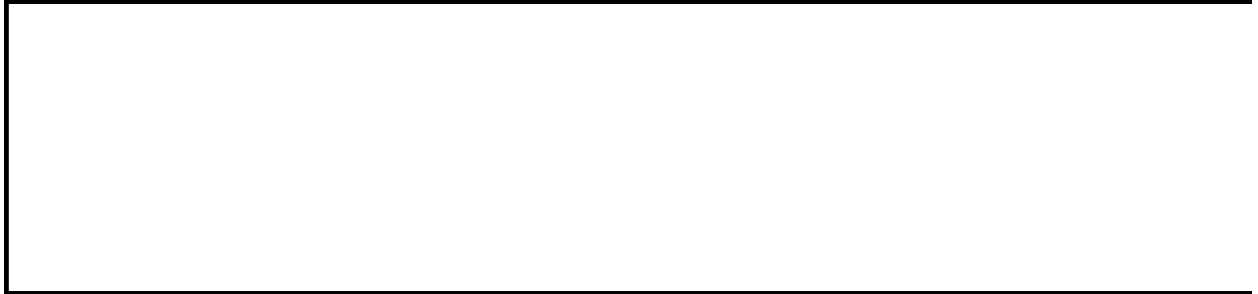
## Remarks

You must specify one, and only one, of these three arguments.

## Example

This example deletes the name "SortRange" from the workbook.

```
Spreadsheet1.ActiveWorkbook.Names("SortRange").Delete
```



# LargeScroll Method

-

Scrolls the contents of the window by pages. The size of the pages is determined by the number of rows and columns visible in the active window.

*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

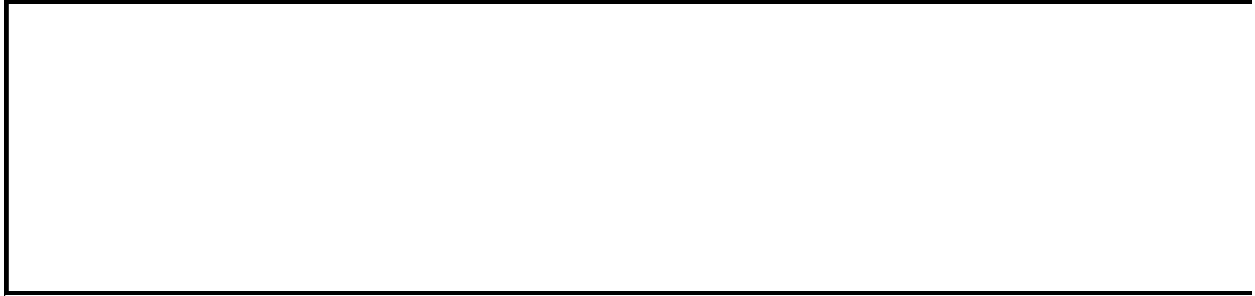
If ***Down*** and ***Up*** or ***ToRight*** and ***ToLeft*** are both specified, the contents of the window are scrolled by the difference of the arguments. For example, if ***Down*** is 3 and ***Up*** is 6, the contents are scrolled up three pages.

Any of the arguments can be a negative number.

## Example

This example scrolls the contents of the active window of Spreadsheet1 down three pages and to the right two pages.

Spreadsheet1.ActiveWindow.**LargeScroll** 3, ,2





# Load Method

-

Loads XML chart data from a URL or local file. The file must consist of only charting XML, and it cannot contain any leading HTML. The string must begin with <script language="XML"> and end with </script>.

*expression*.**Load**(*FileName* As String)

*expression* An expression that returns a **ChartSpace** object.

**FileName** Required **String**. Specifies the file containing XML chart data.

## Example

This example loads the specified XML data file.

```
ChartSpace1.Load "chart.xml"
```



# LoadText Method

-

Loads and parses the specified text file into a worksheet. The contents of the text file are loaded into the worksheet beginning at the specified cell. Existing cell contents will be overwritten.

*expression*.**LoadText**(*File*, *Delimiters*, *ConsecutiveDelimAsOne*, *TextQualifier*)

*expression* An expression that returns a [Range](#) object.

**File** Required **String**. Specifies the name of the text file.

**Delimiters** Optional **String**. Specifies the field delimiters.

**ConsecutiveDelimAsOne** Optional **Boolean**. **True** to have consecutive delimiters considered as a single delimiter. The default value is **False**.

**TextQualifier** Optional **String**. Specifies the text qualifier. The default value is the double quotation mark character.

## Example

This example inserts a tab-delimited text file into the active worksheet of Spreadsheet1. The contents of the text file will begin in cell B10.

```
Spreadsheet1.ActiveSheet.Range("B10").LoadText "tabfile.txt",  
Chr$(9)
```



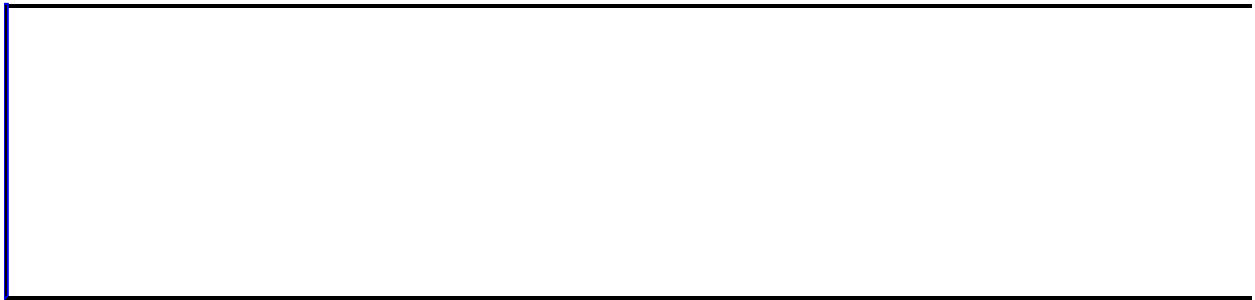
# MakeCurrent Method

Makes the specified section the current section.

*expression*.**MakeCurrent**(*ScrollIntoView*)

*expression* An expression that returns a [Section](#) object.

**ScrollIntoView** Optional **Boolean**. Set this argument to **True** to scroll the section into view. The default value is **False**.



# Merge Method

-

Creates a merged cell from the specified range. When you create a merged cell, the value in the upper-left cell in the specified range is used for the merged cell value. All other cell values in the merged cell are ignored.

*expression*.**Merge**(**Across**)

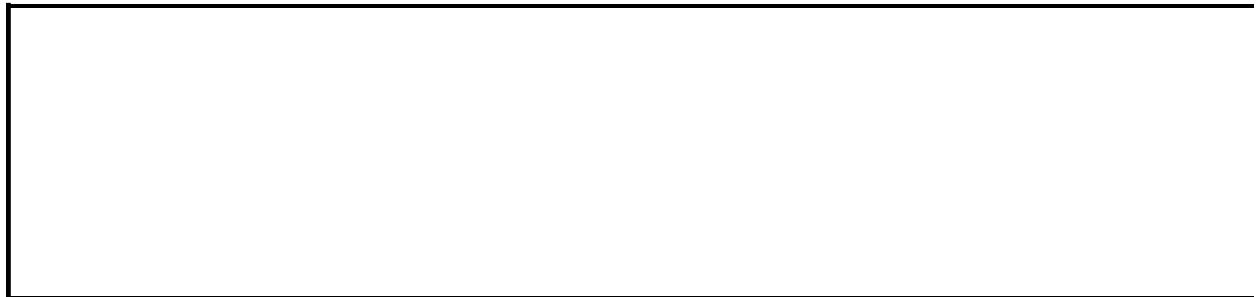
*expression* An expression that returns a [Range](#) object. This method fails if the range only partially encloses a previously merged cell.

**Across** Optional **Variant**. **True** to merge cells in each row in the specified range as separate merged cells. The default value is **False**.

## Example

This example creates a merged cell from the range B2:C5 and puts a thick red border around the merged cell.

```
Sub Merge_Cells()  
    Dim ssConstants  
    Dim rngMerged  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Merge cells B2:C5.  
    Spreadsheet1.ActiveSheet.Range("B2:C5").Merge  
  
    ' Set a variable to the merged range.  
    Set rngMerged = Spreadsheet1.ActiveSheet.Range("B2").MergeArea  
  
    ' Format the merged cell.  
    rngMerged.Borders.Color = "Red"  
    rngMerged.Borders.Weight = ssConstants.owcLineWeightThick  
    rngMerged.HorizontalAlignment = ssConstants.xlHAlignCenter  
    rngMerged.VerticalAlignment = ssConstants.xlVAlignCenter  
End Sub
```



# Move Method

-  
Moves the sheet to another location in the workbook.

*expression*.**Move**(*Before*, *After*)

*expression* Required. An expression that returns one of the objects 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**.





# MoveDetailLeft Method

-

Scrolls the detail area to the left starting at the specified column member.

*expression*.**MoveDetailLeft**(*DetailLeft*, *DetailLeftOffset*, *Update*)

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

***DetailLeft*** Required **Long**. An index number indicating for the cell to use as the basis for scrolling. For example, use a value of 2 to start the scrolling at the third column in the detail area.

***DetailLeftOffset*** Required **Long**. The number of pixels to scroll.

***Update*** Optional **Boolean**. Determines whether or not the display is updated. The default value is **True**.

## Example

This example scrolls the detail area to the left by 10 pixels starting at the second column.

```
Sub ScrollDetailColumns()  
    Dim ptData  
    Dim pmColumnMember  
  
    Set ptData = PivotTable1.ActiveData  
    Set pmColumnMember = ptData.ColumnAxis.Member  
  
    ' Scroll the first column in the detail area  
    ' the left by 10 pixels.  
    pmColumnMember.MoveDetailLeft 1, 10  
End Sub
```



# MoveDetailTop Method

-

Scrolls the the detail area down starting at the specified cell.

*expression*.**MoveDetailTop**(*DetailTop*, *DetailTopOffset*, *Update*)

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

***DetailTop*** Required **Long**. An index number indicating the cell to use as the basis for scrolling. For example, use a value of 5 to start the scrolling at the sixth row in the detail area.

***DetailTopOffset*** Required **Long**. The number of pixels to scroll. Use 0 to scroll the record specified in the ***DetailTop*** argument to the top of the detail area.

***Update*** Optional **Boolean**. Determines whether or not the display is updated. The default value is **True**.

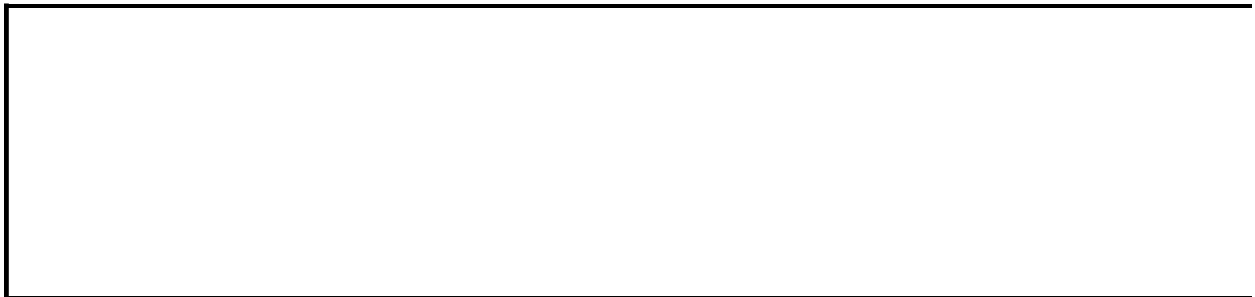
## Remarks

Use the **MoveDetailLeft** method to scroll the detail area to the left.

## Example

This example scrolls the fifteenth row to the first row displayed in the detail area.

```
Sub ScrollDetailArea()  
  
    Dim ptData  
    Dim pmColumnMember  
    Dim pmRowMember  
  
    Set ptData = PivotTable1.ActiveData  
  
    Set pmRowMember = ptData.RowAxis.Member  
    Set pmColumnMember = ptData.ColumnAxis.Member  
  
    ' Starting at the 15th row of the detail area, scroll 100 pixels  
    ptData.Cells(pmRowMember, pmColumnMember).MoveDetailTop 15, 0  
  
End Sub
```



# MoveFirst Method

-

Moves to the first record in the data access page recordset. This method fails if the current record is the first record in the recordset.

*expression*.**MoveFirst**

*expression* An expression that returns a [DataPage](#) object.



# MoveLast Method

-

Moves to the last record in the data access page recordset. This method fails if the current record is the last record in the recordset.

*expression*.**MoveLast**

*expression* An expression that returns a [DataPage](#) object.



# MoveLeft Method

-

Scrolls a column field member left by the specified number of pixels, or until the next row member has been scrolled to the left side of the display.

*expression*.**MoveLeft**(*Left*, *LeftOffset*, *Update*)

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

*Left* Required **PivotColumnMember** object. The column member to scroll.

*LeftOffset* Required **Long**. The number of pixels to scroll the member.

*Update* Optional **Boolean**. Determines whether or not the display is updated. The default value is **True**.



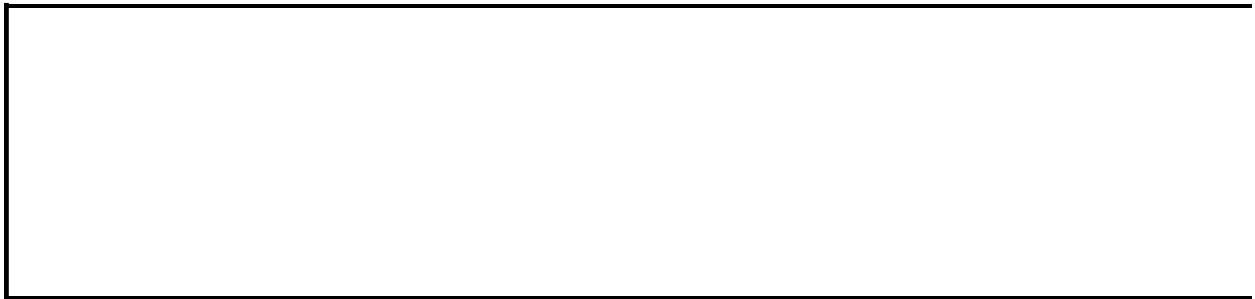
## Remarks

Use the **MoveTop** property to scroll row field members.

## Example

This example scrolls the PivotTable view to the next column member.

```
Sub ScrolltoNextColumnMember()  
    Dim ptData  
    Set ptConstants = PivotTable1.Constants  
    Set ptData = PivotTable1.ActiveData  
    ' Scroll to the next column member.  
    ptData.MoveLeft ptData.Left, 1000  
End Sub
```



# MoveNext Method

-

Moves to the next record in the data access page recordset. This method fails if the current record is the last record in the recordset.

*expression*.**MoveNext**

*expression* An expression that returns a [DataPage](#) object.



# MovePrevious Method

-

Moves to the previous record in the data access page recordset. This method fails if the current record is the first record in the recordset.

*expression*.**MovePrevious**

*expression* An expression that returns a [DataPage](#) object.



# MoveTop Method

-

Scrolls a row field member up by the specified number of pixels, or until the next row member has been scrolled to the top of the display.

*expression*.**MoveTop**(*Top*, *TopOffset*, *Update*)

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

**Top** Required **PivotRowMember** object. The row member to scroll.

**TopOffset** Required **Long**. The number of pixels to scroll the member.

**Update** Optional **Boolean**. Determines whether or not the display is updated. The default value is **True**.

## Remarks

Use the **MoveLeft** property to scroll column field members.

## Example

This example scrolls the PivotTable view to the next row member.

```
Sub ScrolltoNextRowMember()  
    Dim ptData  
    Set ptConstants = PivotTable1.Constants  
    Set ptData = PivotTable1.ActiveData  
    ' Scroll to the next row member.  
    ptData.MoveTop ptData.Top, 1000  
End Sub
```



# NewRecord Method

-

Adds a new record to the data access page recordset. This method fails if the recordset cannot be updated.

*expression*.**NewRecord**

*expression* An expression that returns a [DataPage](#) object.





# Nz Method

-

Use this method to return zero, a zero-length string (" "), or another specified value when a value is **Null**. For example, you can use this function to convert a **Null** value to another value and prevent it from propagating through an expression. Returns a **Variant**.

*expression.Nz(Value, ValueIfNull)*

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

**Value** Required **Variant**. The value to convert.

**ValueIfNull** Optional **Variant**. Value to return if the **Value** argument argument is **Null**. This argument enables you to return a value other than zero or a zero-length string.

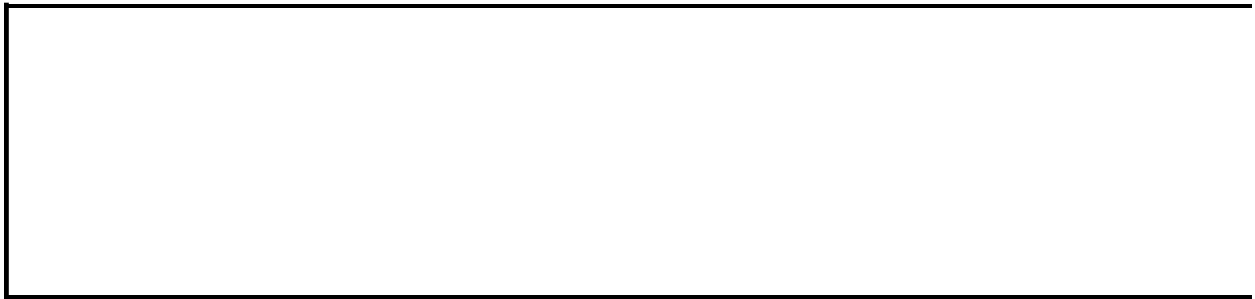
## Remarks

This method is useful for expressions that may include **Null** values. To force an expression to evaluate to a non-**Null** value even when it contains a **Null** value, use the this method to return a zero, a zero-length string, or a custom return value.

For example, the expression `2 + varX` will always return a **Null** value when the **Variant** `varX` is **Null**. However, `2 + MSODSC.Nz(varX)` returns 2.

In the next example, the optional argument supplied to the **Nz** method provides the string to be returned if `varFreight` is **Null**.

```
varResult = MSODSC.Nz(varFreight, "No Freight Charge")
```



# ParseText Method

-  
Parses the specified text string and places the result in the specified range.

*expression*.**ParseText**(*Text*, *Delimiters*, *ConsecutiveDelimAsOne*, *TextQualifier*)

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

**Text** Required **String**. Specifies the string to be parsed.

**Delimiters** Optional **String**. Specifies the field delimiters.

**ConsecutiveDelimAsOne** Optional **Boolean**. **True** to have consecutive delimiters considered as one delimiter. The default value is **False**.

**TextQualifier** Optional **String**. Specifies the text qualifier. The default value is the double quotation mark character.

## Example

This example parses the specified string into the range starting at cell A1.

`Spreadsheet1.ActiveSheet.Range("A1").ParseText "name, address, city,`

--

▼ [Show All](#)

# Paste Method

-

## ► [Paste method as it applies to the \*\*Worksheet\*\* object.](#)

Pastes the contents of the Clipboard onto the sheet. If you don't specify the **Destination** argument, you must select the destination range before you use this method.

*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.

**Link** Optional **Variant**. Not supported in this version of the Spreadsheet component.

## ► [Paste method as it applies to the \*\*Range\*\* object.](#)

Pastes the Clipboard contents into the specified range.

*expression*.**Paste**

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

## Example

### ► [As it applies to the \*\*Worksheet\*\* object.](#)

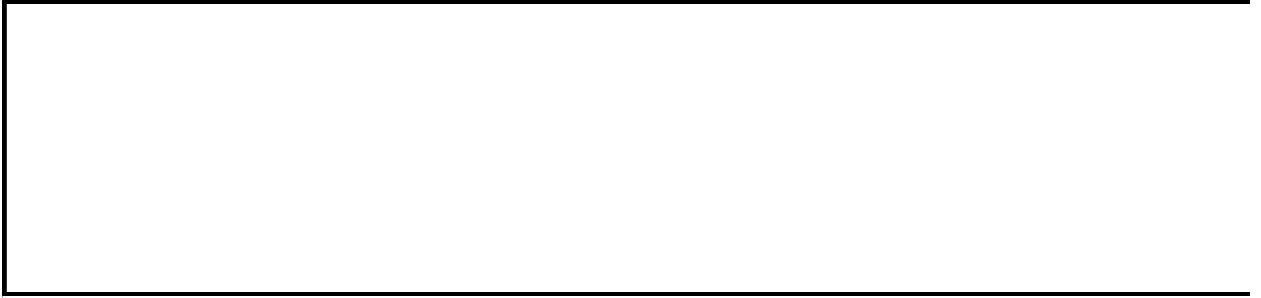
This example copies cells A1:D10 on Sheet1 and pastes starting at cell A1 in Sheet2.

```
Sub PasteData()  
  
    ' Copy cell A1:D10 in Sheet1.  
    Spreadsheet1.Sheets("Sheet1").Range("A1:D10").Copy  
  
    ' Paste starting at cell A1 of Sheet2.  
    Spreadsheet1.ActiveSheet.Paste  
    Spreadsheet1.Sheets("Sheet2").Range("A1")  
  
End Sub
```

### ► [As it applies to the \*\*Range\*\* object.](#)

This example pastes the contents of cell A1 into cells A3, A5, and A7.

```
Sub PasteRange()  
  
    Dim iRowNum  
  
    ' Copy cell A1 on the active worksheet.  
    Spreadsheet1.ActiveSheet.Range("A1").Copy  
  
    ' Loop through the odd numbers from 3 to 7.  
    For iRowNum = 3 To 7 Step 2  
        ' Paste the copied cell.  
        Spreadsheet1.ActiveSheet.Cells(iRowNum, 1).Paste  
    Next  
  
End Sub
```





# PointsToScreenPixelsX Method

-

Returns a **Long** value that represents the number of pixels from the left side of the spreadsheet's window to the left side the first column in the spreadsheet, plus the value specified in the ***Points*** argument.

*expression*.**PointsToScreenPixelsX**(*Points*)

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

*Points* Required **Long**. The number of pixels to add to this method's result.



# PointsToScreenPixelsY Method

-

Returns a **Long** value that represents the number of pixels from the top of the spreadsheet's window to the top of the first row in the spreadsheet, plus the value specified in the **Points** argument.

*expression*.**PointsToScreenPixelsY**(**Points**)

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

**Points** Required **Long**. The number of pixels to add to this method's result.



▼ [Show All](#)

# Protect Method

► [Protect method as it applies to the \*\*Worksheet\*\* object.](#)

Protects a worksheet from modification. This method is equivalent in functionality to using the properties of the [Protection](#) object to protect your worksheet.

*expression*.**Protect**(*Password*, *DrawingObjects*, *Contents*, *Scenarios*, *UserInterfaceOnly*, *AllowFormattingCells*, *AllowFormattingColumns*, *AllowFormattingRows*, *AllowInsertingColumns*, *AllowInsertingRows*, *AllowInsertingHyperlinks*, *AllowDeletingColumns*, *AllowDeletingRows*, *AllowSorting*, *AllowFiltering*, *AllowUsingPivotTableReports*)

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

**Password** Optional **Variant**. This argument is not supported.

**DrawingObjects** Optional **Variant**. This argument is not supported.

**Contents** Optional **Variant**. **True** to protect contents of the locked cells on the worksheet. The default value is **True**.

**Scenarios** Optional **Variant**. This argument is not supported.

**UserInterfaceOnly** Optional **Variant**. This argument is not supported.

**AllowFormattingCells** Optional **Variant**. This argument is not supported.

**AllowFormattingColumns** Optional **Variant**. **True** to allow the resizing of columns. The default value is **False**. Equivalent to the **AllowSizingAllColumns** property of the **Protection** object.

**AllowFormattingRows** Optional **Variant**. **True** to allow the resizing of rows. The default value is **False**. Equivalent to the **AllowSizingAllRows** property of the **Protection** object.

***AllowInsertingColumns*** Optional **Variant**. **True** if worksheet columns can be inserted. The default value is **False**. Equivalent to the **AllowInsertingColumns** property of the **Protection** object.

***AllowInsertingRows*** Optional **Variant**. **True** if worksheet rows can be inserted. The default value is **False**. Equivalent to the **AllowInsertingRows** property of the **Protection** object.

***AllowInsertingHyperlinks*** Optional **Variant**. This argument is not supported.

***AllowDeletingColumns*** Optional **Variant**. **True** if worksheet columns can be deleted. The default value is **False**. Equivalent to the **AllowDeletingColumns** property of the **Protection** object.

***AllowDeletingRows*** Optional **Variant**. **True** if worksheet rows can be deleted. The default value is **False**. Equivalent to the **AllowDeletingRows** property of the **Protection** object.

***AllowSorting*** Optional **Variant**. **True** if the specified worksheet can be sorted. The default value is **False**. Equivalent to the **AllowSorting** property of the **Protection** object.

***AllowFiltering*** Optional **Variant**. **True** if the specified worksheet can be filtered. The default value is **False**. Equivalent to the **EnableAutoFilter** property of the **Protection** object.

***AllowUsingPivotTableReports*** Optional **Variant**. This argument is not supported.

► [Protect method as it applies to the \*\*Workbook\*\* object.](#)

Protects a workbook so that it cannot be modified.

*expression*.**Protect**(*Password*, *Structure*, *Windows*)

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

***Password*** Optional **Variant**. This argument is not supported.

***Structure*** Optional **Variant**. **True** to protect the structure of the workbook.

Protecting the structure of the workbook prevents sheets from being moved, inserted, deleted, hidden, unhidden and renamed. The default value is **False**.

**Windows** Optional **Variant**. This argument is not supported.

## Remarks

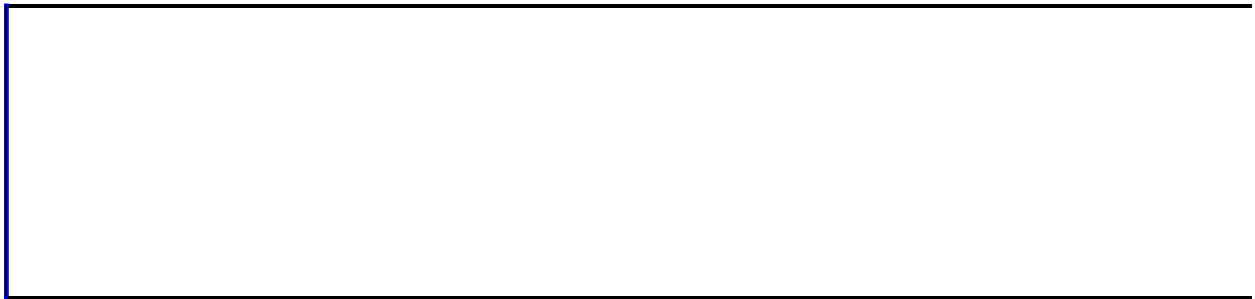
Setting an unsupported argument to **True** will result in an run-time error.

## Example

► [As it applies to the \*\*Worksheet\*\* object.](#)

This example protects the contents of the locked cells on Sheet1 in Spreadsheet1 while allowing the user to insert and delete columns and rows.

```
Spreadsheet1.Worksheets("Sheet1").Protect , , True, , , , , True,
```





▼ [Show All](#)

# RangeFromPoint Method

▸ [RangeFromPoint method as it applies to the \*\*ChartSpace\*\* object.](#)

Returns the most detailed object at the specified location in a chart workspace. For example, if you specify the x- and y-coordinates of a point, the corresponding **ChPoint** object is returned, even though a series also exists at those coordinates.

*expression*.**RangeFromPoint**(*X*, *Y*)

*expression* Required. An expression that returns a [ChartSpace](#) object.

*X* Required **Long**.

*Y* Required **Long**.

▸ [RangeFromPoint method as it applies to the \*\*Window\*\* object.](#)

Returns the **Range** object that is positioned at the specified pair of screen coordinates. If there isn't a range located at the specified coordinates, this method returns **Nothing**.

*expression*.**RangeFromPoint**(*X*, *Y*)

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

*X* Required **Long**.

*Y* Required **Long**.

## Example

► [As it applies to the \*\*ChartSpace\*\* object.](#)

This example changes the value of an HTML text box named "Textbox" when the mouse pointer passes over an object in ChartSpace1. The value of the text box reflects the type of object that the mouse pointer passes.

```
Sub ChartSpace1_MouseMove(ByVal Button, ByVal Shift, ByVal x, ByVal  
    ' Set the value of Textbox to the type of object .  
    Textbox.Value = TypeName(ChartSpace1.RangeFromPoint(x,y))  
End Sub
```



# RectIntersect Method

-

Returns a **Range** object that represents the rectangular intersection of the specified ranges. Returns **Nothing** if the specified ranges do not overlap.

*expression*.**RectIntersect**(*Range1*, *Range2*)

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

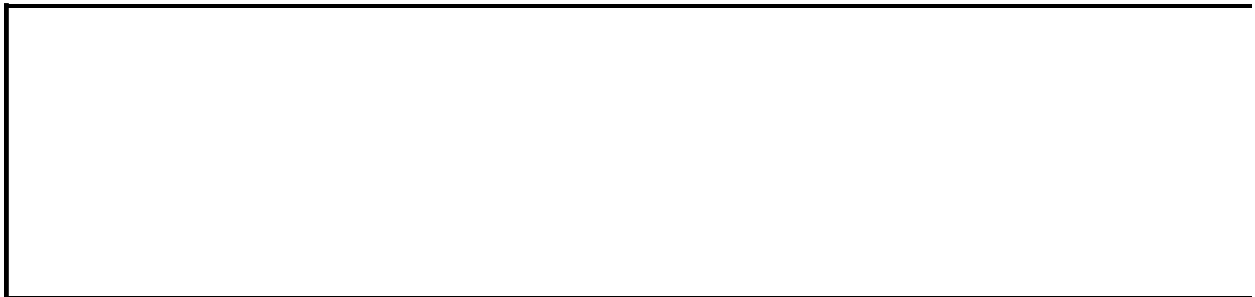
*Range1* Required **Range**.

*Range2* Required **Range**.

## Example

This example bolds the cells where the named range "Range1" overlaps the named range "Range2" in the active sheet of Spreadsheet1.

```
Sub BoldIntersection()  
  
    Dim rngIntersect  
    Dim rngFirstRange  
    Dim rngSecondRange  
  
    ' Set a variable to the first named range.  
    Set rngFirstRange = Spreadsheet1.ActiveSheet.Range("Range1")  
  
    ' Set a variable to the second named range.  
    Set rngSecondRange = Spreadsheet1.ActiveSheet.Range("Range2")  
  
    ' Set a variable to the intersection of the two named ranges.  
    Set rngIntersect = Spreadsheet1.RectIntersect(rngFirstRange, rngSecondRange)  
  
    ' Check whether the named ranges overlap.  
    If Not rngIntersect Is Nothing Then  
        ' Bold the font in the overlapping portion  
        ' of the two ranges.  
        rngIntersect.Font.Bold = True  
    End If  
End Sub
```



# RectUnion Method

Returns a **Range** object that represents the smallest range of cells that includes the union of the specified ranges.

*expression*.**RectUnion**(*Range1*, *Range2*)

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

*Range1* Required **Range**.

*Range2* Required **Range**.

## Remarks

This method always returns a rectangular range. For example, if you specify A1:A5 and F1:F10, the return value is the rectangular range A1:F10. Also, you cannot use this method to create a range containing noncontiguous areas.

# Refresh Method

-  
Refreshes the specified object (reloads source data or repaints the object).

*expression*.**Refresh**

*expression* An expression that returns an object in the Applies To list.



## Example

This example causes the chart workspace to repaint itself.

`ChartSpace1.Refresh`

This example causes a data source control-bound chart to reload its data.

`ChartSpace1.ChartDataSources(0).Refresh`

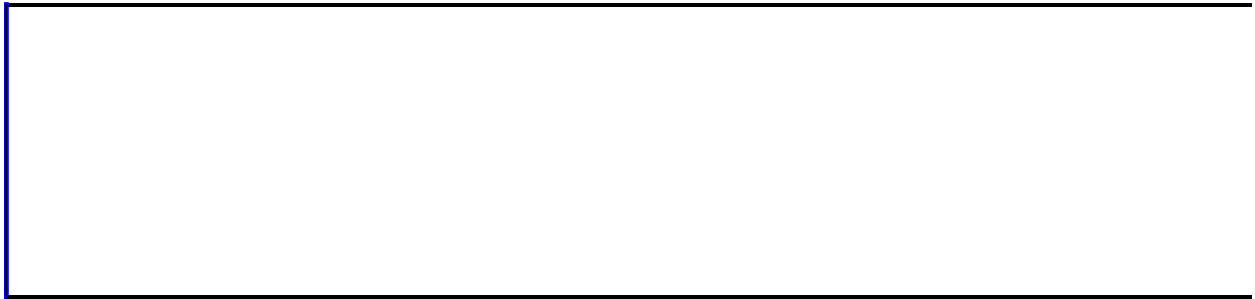


# RefreshJetCache Method

-  
Refreshes the data access page's connection with a Microsoft Access database.

*expression*.**RefreshJetCache**( )

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



# RemoveFieldSet Method

-  
Removes a field set from the specified axis.

*expression*.**RemoveFieldSet**(*FieldSet*)

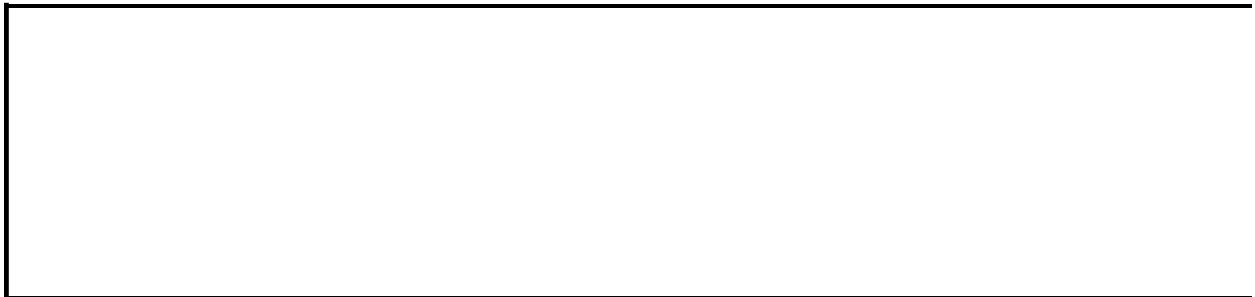
*expression* An expression that returns a [PivotDataAxis](#), [PivotFilterAxis](#), or [PivotGroupAxis](#) object.

**FieldSet** Required **Variant**. Specifies the field set to be removed. Can be a [PivotFieldSet](#) object, a field set name, or a field set index number.

## Example

This example removes the ShipVia field from the filter axis in PivotTable1.

```
PivotTable1.ActiveView.FilterAxis.RemoveFieldSet "ShipVia"
```



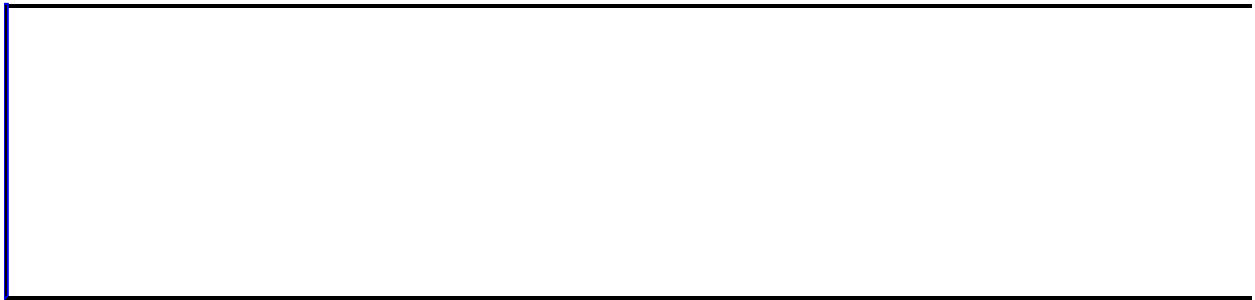
# RemoveTotal Method

-  
Removes a total from the specified data axis.

*expression*.**RemoveTotal**(*Total*)

*expression* An expression that returns a [PivotDataAxis](#) object.

**Total** Required **Variant**. Specifies the total to be removed. Can be a [PivotTotal](#) object, a total's name, or a total's index number.



## Example

This example removes the Order Count total from the current view in PivotTable1.

```
PivotTable1.ActiveView.DataAxis.RemoveTotal "Order Count"
```

# Repaint Method

-  
Forces a redraw of the specified object.

*expression*.**Repaint**

*expression* Required. An expression that returns a **ChartSpace** or **Spreadsheet** object.



# Requery Method

-  
Executes the query that returned the recordset and all related recordsets for the specified object.

*expression*.**Requery**

*expression* An expression that returns a **DataPage** object.





# Reset Method

-

Resets the specified **Heading** object. Use this method to reset a specific row or column heading. Use the **ResetHeadings** method if you want to reset all row and column headings in a window to their default values.

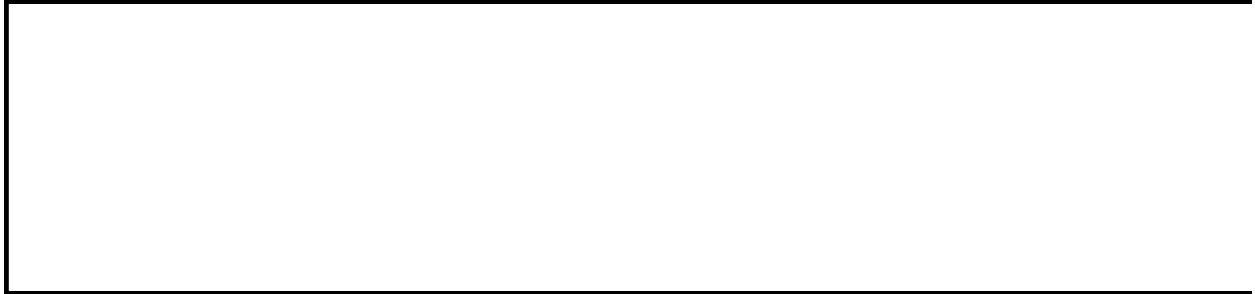
*expression*.**Reset**

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

## Example

This example resets the caption of column D in the active window of Spreadsheet1 to its default value.

```
Spreadsheet1.ActiveWindow.ColumnHeadings(4).Reset
```



# ResetColors Method

-  
Resets the color palette of the specified workbook to the default colors.

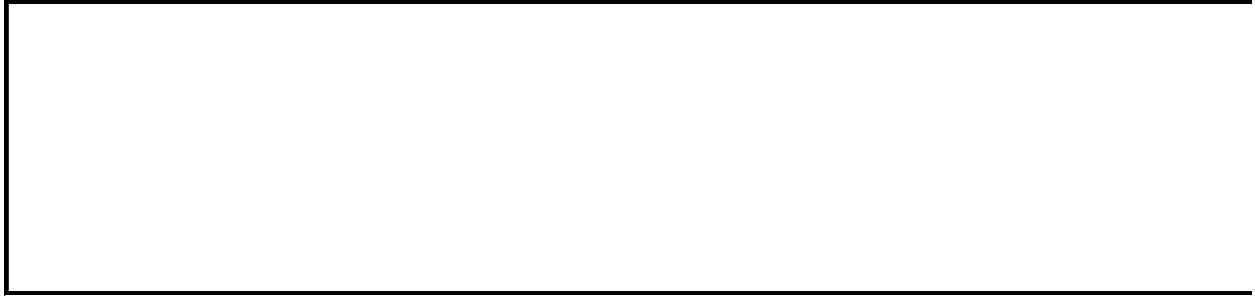
*expression*.**ResetColors**

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

## Example

The following example resets the color palette of the workbook that is open in Spreadsheet1 back to its default setting.

Spreadsheet1.ActiveWorkbook.**ResetColors**



# ResetHeadings Method

-

Resets the row and columns headings of the specified window to their default values. Use the **Reset** method instead if you want to reset specific row and column headings.

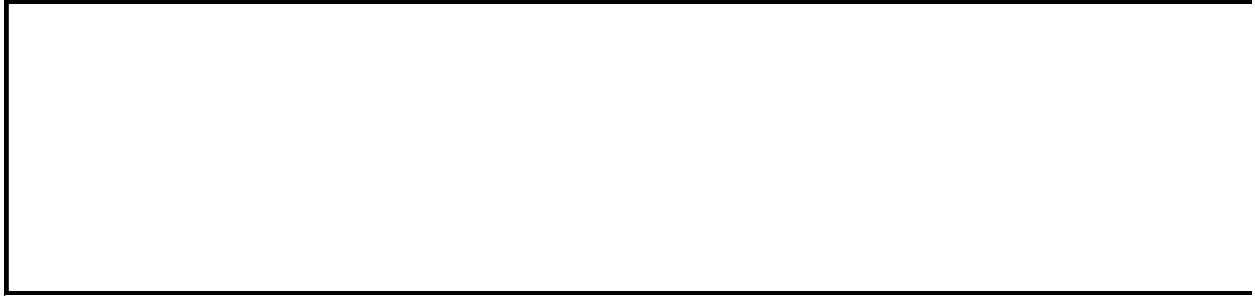
*expression*.**ResetHeadings**

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

## Example

The following example resets the row and column headings of the active window in Spreadsheet1 to their default values.

Spreadsheet1.ActiveWindow.**ResetHeadings**



# RotateClockwise Method

-  
Rotates all series in the specified chart clockwise in 90-degree increments.

*expression*.**RotateClockwise**

*expression* An expression that returns a **ChPlotArea** object.

## Example

This example rotates all series in the specified chart clockwise 90 degrees.

```
ChartSpace1.Charts(0).PlotArea.RotateClockwise
```





# RotateCounterClockwise Method

-  
Rotates all series in the specified chart counterclockwise in 90-degree increments.

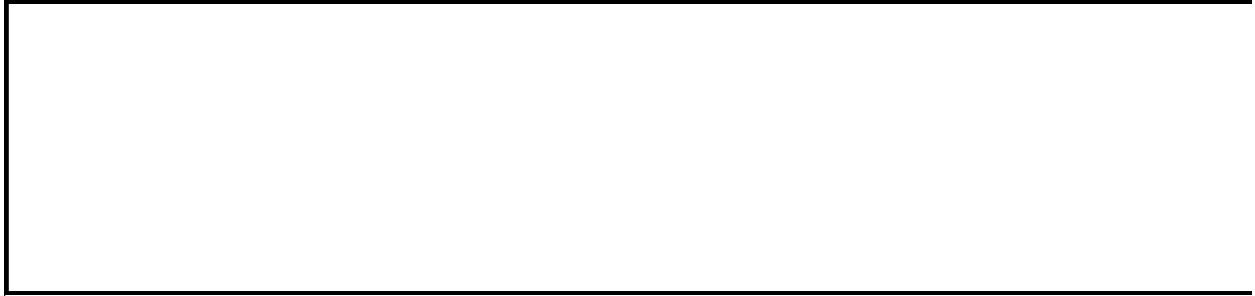
*expression*.**RotateCounterClockwise**

*expression* An expression that returns a **ChPlotArea** object.

## Example

This example rotates all series in the specified chart counterclockwise 90 degrees.

```
ChartSpace1.Charts(0).PlotArea.RotateCounterClockwise
```



# Save Method

-

Saves the current record to the database associated with the specified data access page.

*expression*.**Save**

*expression* An expression that returns a [DataPage](#) object.



# ScrollIntoView Method

-

Scrolls the document window so that the contents of a specified rectangular area are displayed in either the upper-left or lower-right corner of the document window or pane (depending on the value of the **Start** argument).

*expression*.**ScrollIntoView**(**Left**, **Top**, **Width**, **Height**, **Start**)

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

**Left** Required **Long**. The horizontal position of the rectangle (in points) from the left edge of the document window or pane.

**Top** Required **Long**. The vertical position of the rectangle (in points) from the top of the document window or pane.

**Width** Required **Long**. The width of the rectangle in points.

**Height** Required **Long**. The height of the rectangle in points.

**Start** Optional **Variant**. **True** to have the upper-left corner of the rectangle appear in the upper-left corner of the document window or pane. **False** to have the lower-right corner of the rectangle appear in the lower-right corner of the document window or pane. The default value is **True**.

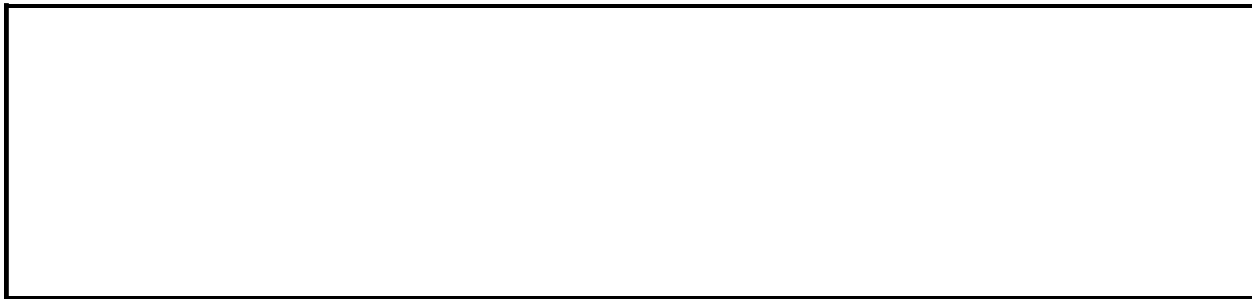
## Remarks

The ***Start*** argument is useful for orienting the screen display when the rectangle is larger than the document window.

## Example

This example defines a 100-by-200-pixel rectangle in the active document window, positioned 20 pixels from the top of the window and 50 pixels from the left edge of the window. The example then scrolls the document up and to the left so that the upper-left corner of the rectangle is aligned with the upper-left corner of the window.

```
Spreadsheet1.ActiveWindow.ScrollIntoView 50, 20, 100, 200
```



▼ [Show All](#)

# Select Method

▸ [Select method as it applies to the \*\*ChartSpace\*\*, \*\*ChAxis\*\*, \*\*ChCategoryLabel\*\*, \*\*ChChart\*\*, \*\*ChChartField\*\*, \*\*ChDataLabel\*\*, \*\*ChDataLabels\*\*, \*\*ChErrorBars\*\*, \*\*ChGridlines\*\*, \*\*ChLegend\*\*, \*\*ChLegendEntry\*\*, \*\*ChPlotArea\*\*, \*\*ChPoint\*\*, \*\*ChSeries\*\*, \*\*ChSurface\*\*, \*\*ChTitle\*\*, \*\*ChTrendline\*\*, and \*\*Range\*\* objects.](#)

Selects the specified object.

*expression*.**Select**

*expression* Required. An expression that returns one of the above objects.

▸ [Select method as it applies to the \*\*Sheets\*\*, \*\*Worksheet\*\*, and \*\*Worksheets\*\* objects.](#)

Selects the specified sheet.

*expression*.**Select**(**Replace**)

*expression* Required. An expression that returns one of the above objects.

**Replace** Optional **Variant**. **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.

▸ [Select method as it applies to the \*\*PivotTable\*\* object.](#)

Selects an object in a PivotTable list.

*expression*.**Select**(**Selection**, **ActiveObject**, **ScrollType**, **Update**, **Notify**)

*expression* Required. An expression that returns one of the above objects.



***Selection*** Required **Object**. The object to select.

***ActiveObject*** Required **Object**. The object to make active within the new selection.

***ScrollType*** Optional [PivotScrollTypeEnum](#). Specifies the method to use to scroll the new selection so that it is visible.

PivotScrollTypeEnum can be one of these PivotScrollTypeEnum constants.

**plScrollTypeAll**

**plScrollTypeBottom**

**plScrollTypeLeft**

**plScrollTypeNone** *default*

**plScrollTypeRight**

**plScrollTypeTop**

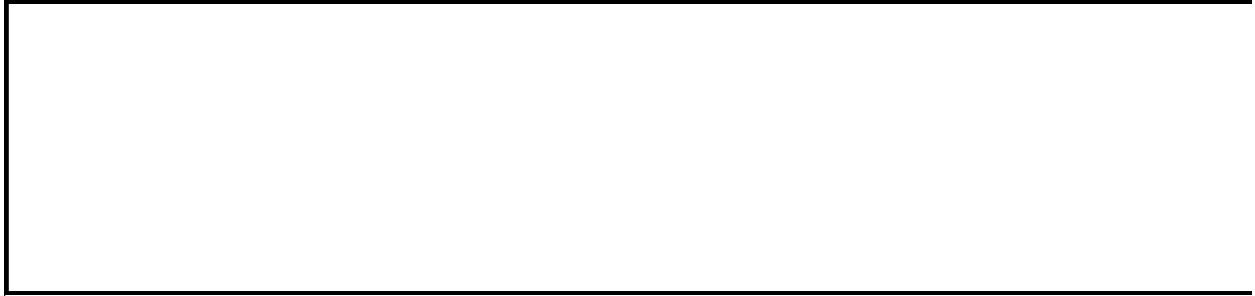
***Update*** Optional **Boolean**. Determines whether the display is updated immediately. The default value is **True**.

***Notify*** Optional **Boolean**. This argument is not supported.

## Example

This example selects the cell that is one column to the right of and in the same row as the active cell.

```
Spreadsheet1.ActiveCell.Offset(0, 1).Select
```



▼ [Show All](#)

# SetData Method

Sets data for the specified chart object.

*expression.SetData(Dimension, DataSourceIndex, DataReference)*

*expression* An expression that returns a **ChChart**, **ChErrorBars** **ChSeries**, or **ChartSpace** object.

**Dimension** Required [ChartDimensionsEnum](#) constant. Specifies the data dimension to be set.

ChartDimensionsEnum can be one of these ChartDimensionsEnum constants.

**chDimBubbleValues** Set the values for the markers on a Bubble chart.

**chDimCategories** Set the values to use as categories.

**chDimCharts** Sets the source fields for new charts when the **HasMultipleCharts** property is set to **True**.

**chDimCloseValues** Set the closing values for a Stock chart.

**chDimFilter** Sets the fields to place on the filter axis.

**chDimFormatValues** Set the values to use in a format map.

**chDimHighValues** Set the high values for a Stock chart.

**chDimLowValues** Set the low values for a Stock chart.

**chDimOpenValues** Set the opening values for a Stock chart.

**chDimRValues** Set the R values for a Polar chart.

**chDimSeriesNames** Set the values to use as series names.

**chDimThetaValues** Set the Theta values for a Polar chart.

**chDimValues** Set the values to be charted.

**chDimXValues** Set the x values for an XY (Scatter) or Bubble chart.

**chDimYValues** Set the y values for an XY (Scatter) or Bubble chart.

**DataSourceIndex** Required **Long**. Can be a [ChartSpecialDataSourcesEnum](#) constant.

ChartSpecialDataSourcesEnum can be one of these ChartSpecialDataSourcesEnum constants.

**chDataBound** Binds the specified object to the external data source specified in the *DataReference* argument.

**chDataLinked** Binds the specified object to another dimension. Use this value when you specify **chDimFormatValues** in the *Dimension* argument to create a format map.

**chDataLiteral** Binds the specified object to the literal data specified in the *DataReference* argument.

**chDataNone** Clears the specified object.

*DataReference* Optional **Variant**. For ChChart and ChSeries objects, this argument specifies the data reference as a Microsoft Excel-style range reference ("A1:D4" , for example), or a row-set column name. When the

*DataSourceIndex* argument is set to **chDataLiteral**, you can set *DataReference* to a one-dimensional array or a comma-delimited list. For *ChErrorBars* objects, this argument specifies an array of **Double** or **String** values you can use for error-bar values. Note that you can use this argument only with custom error bars (the error-bar [Type](#) property must be set to **chErrorBarTypeCustom**).

## Remarks

Alternatively, you can specify a [ChartErrorBarCustomValuesEnum](#) constant for the ***Dimension*** argument to specify the values to use for error bars.

ChartErrorBarCustomValuesEnum can be one of these ChartErrorBarCustomValuesEnum constants.

**chErrorBarMinusValues**

**chErrorBarPlusValues**

You can bind a chart to only one data source. For example, if you have two charts in a ChartSpace, you cannot bind them to different data sources. However, you can bind a chart or data series to a set of literal data once the chart or ChartSpace has been bound to an external data source.

When binding to an OLAP data source, the ***DataReference*** argument can bind to a field set, but not a field. You can pass an array of fields to the ***DataReference*** argument to bind to a specific field or fields when connected to an OLAP data source.

## Example

This example creates a chart using literal data arrays.

```
Sub BindChartToArrays()  
  
    Dim asSeriesNames(1)  
    Dim asCategories(7)  
    Dim aiValues(7)  
    Dim chConstants  
    Dim chtNewChart  
  
    asSeriesNames(0) = "Satisfaction Data"  
  
    asCategories(0) = "Very Good"  
    asCategories(1) = "Good"  
    asCategories(2) = "N/A"  
    asCategories(3) = "Average"  
    asCategories(4) = "No Response"  
    asCategories(5) = "Poor"  
    asCategories(6) = "Very Poor"  
  
    aiValues(0) = 10  
    aiValues(1) = 22  
    aiValues(2) = 6  
    aiValues(3) = 31  
    aiValues(4) = 5  
    aiValues(5) = 14  
    aiValues(6) = 12  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Add a new chart to Chartspace1.  
    Set chtNewChart = ChartSpace1.Charts.Add  
  
    ' Specify that the chart is a column chart.  
    chtNewChart.Type = chConstants.chChartTypeColumnClustered  
  
    ' Bind the chart to the arrays.  
    chtNewChart.SetData chConstants.chDimSeriesNames, chConstants.ch  
    chtNewChart.SetData chConstants.chDimCategories, chConstants.chD  
    chtNewChart.SeriesCollection(0).SetData chConstants.chDimValues,  
  
End Sub
```

This example creates a chart that is bound to a spreadsheet. The series name is in

cell B1, the category names are in cells A2:A28, and the values are in cells B2:B28.

```
Sub BindToSpreadsheet()  
    Dim chConstants  
    Dim chtChart1  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set the data source of ChartSpace1 to Spreadsheet1.  
    Set ChartSpace1.DataSource = Spreadsheet1  
  
    ' Set a variable to a new chart in Chartspace1.  
    Set chtChart1 = ChartSpace1.Charts.Add  
  
    ' Set the chart type.  
    chtChart1.Type = chConstants.chChartTypeLineMarkers  
  
    ' Bind the series name to cell B1 in the first sheet of Spreadsh  
    chtChart1.SetData chConstants.chDimSeriesNames, chConstants.chDa  
  
    ' Bind the category axis to cell A2:A28 in the first sheet of Sp  
    chtChart1.SetData chConstants.chDimCategories, chConstants.chDat  
  
    ' Bind the values of the data series to cells B2:B28 in the first  
    chtChart1.SeriesCollection(0).SetData chConstants.chDimValues, c  
  
End Sub
```

The following example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created. The smaller values are displayed in white, then larger values are displayed in a light shade of blue, and finally the largest values in the chart are displayed in dark blue.

```
Sub Window_Onload()  
  
    Dim serSeries1  
    Dim segSegment1  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=Se  
    ChartSpace1.DataMember = "Order Details"
```



```

' The following two lines of code bind ChartSpace1 to the Quanti
' in the Order details table.
ChartSpace1.SetData chConstants.chDimCategories, chConstants.chD
ChartSpace1.SetData chConstants.chDimValues, chConstants.chDataB

' Create a format map.
ChartSpace1.SetData chConstants.chDimFormatValues, chConstants.c

' Set a variable to the first series in the first chart in Chart
Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)

' Add a segment to the format map.
Set segSegment1 = serSeries1.FormatMap.Segments.Add

' Specify that the divisions in formatting be created automatica
segSegment1.HasAutoDivisions = True

' Measure the segment boundaries based upon a percentage.
segSegment1.Begin.ValueType = chConstants.chBoundaryValuePercent
segSegment1.End.ValueType = chConstants.chBoundaryValuePercent

' Set the beginning value to 0%, and the ending value to 100%.
segSegment1.Begin.Value = 0
segSegment1.End.Value = 1

' Format the interior of the matching values.
segSegment1.Begin.Interior.Color = "White"
segSegment1.End.Interior.Color = "Blue"

```

End Sub



▼ [Show All](#)

# SetOneColorGradient Method

-  
Fills the specified **ChInterior** object with a one-color gradient.

*expression.SetOneColorGradient(GradientStyle, GradientVariant, GradientDegree, Color)*

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

**GradientStyle** Required [ChartGradientStyleEnum](#). The gradient style.

ChartGradientStyleEnum can be one of these ChartGradientStyleEnum constants.

**chGradientDiagonalDown**

**chGradientDiagonalUp**

**chGradientFromCenter** When setting this argument to **chGradientFromCenter**, the only **GradientVariant** choices are **chGradientVariantEnd** and **chGradientVariantStart**.

**chGradientFromCorner**

**chGradientHorizontal**

**chGradientVertical**

**GradientVariant** Required [ChartGradientVariantEnum](#). The gradient variant.

ChartGradientVariantEnum can be one of these ChartGradientVariantEnum constants.

**chGradientVariantCenter**

**chGradientVariantEdges**

**chGradientVariantEnd**

**chGradientVariantStart**

**GradientDegree** Required **Double**. The gradient degree. Can be a value from 0.0 (dark) through 1.0 (light).

**Color** Optional **Variant**. The foreground color for the gradient. You can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (for example, red is RGB(255, 0, 0)). If this argument is omitted, then the **Color** property is used.

## Example

This example sets the interior fill of the first two series and the plot area of the first chart in ChartSpace1.

```
Sub FormatInteriorColors()  
  
    Dim chConstants  
    Dim serSeries1  
    Dim serSeries2  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
    Set serSeries2 = ChartSpace1.Charts(0).SeriesCollection(1)  
  
    ' Set the interior fill of the first series to a one-color gradient  
    serSeries1.Interior.SetOneColorGradient chConstants.chGradientDi  
        chConstants.chGradientVariantCenter, 0.2, "Blue"  
  
    ' Set the interior fill of the second series to a preset gradient  
    serSeries2.Interior.SetPresetGradient chConstants.chGradientFrom  
        chConstants.chGradientVariantEnd, chConstants.chGradi  
  
    ' Set the interior fill of the plot area to a pattern.  
    ChartSpace1.Charts(0).PlotArea.Interior.SetPatterned chConstants  
        "Yellow", "Blue"  
  
End Sub
```



▼ [Show All](#)

# SetPatterned Method

-  
Fills the specified **ChInterior** object with a preset pattern.

*expression*.**SetPatterned**(*patternType*, *Color*, *BackColor*)

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

*patternType* Required [ChartPatternTypeEnum](#). The pattern style.

ChartPatternTypeEnum can be one of these ChartPatternTypeEnum constants.

**chPattern10Percent**

**chPattern20Percent**

**chPattern25Percent**

**chPattern30Percent**

**chPattern40Percent**

**chPattern50Percent**

**chPattern5Percent**

**chPattern60Percent**

**chPattern70Percent**

**chPattern75Percent**

**chPattern80Percent**

**chPattern90Percent**

**chPatternDarkDownwardDiagonal**

**chPatternDarkHorizontal**

**chPatternDarkUpwardDiagonal**

**chPatternDarkVertical**

**chPatternDashedDownwardDiagonal**

**chPatternDashedHorizontal**

**chPatternDashedUpwardDiagonal**

**chPatternDashedVertical**

chPatternDiagonalBrick  
chPatternDivot  
chPatternDottedDiamond  
chPatternDottedGrid  
chPatternHorizontalBrick  
chPatternLargeCheckerBoard  
chPatternLargeConfetti  
chPatternLargeGrid  
chPatternLightDownwardDiagonal  
chPatternLightHorizontal  
chPatternLightUpwardDiagonal  
chPatternLightVertical  
chPatternNarrowHorizontal  
chPatternNarrowVertical  
chPatternOutlinedDiamond  
chPatternPlaid  
chPatternShingle  
chPatternSmallCheckerBoard  
chPatternSmallConfetti  
chPatternSmallGrid  
chPatternSolidDiamond  
chPatternSphere  
chPatternTrellis  
chPatternWave  
chPatternWeave  
chPatternWideDownwardDiagonal  
chPatternWideUpwardDiagonal  
chPatternZigZag

**Color** Optional **Variant**. The foreground color for the pattern. You can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (for example, red is RGB(255, 0, 0)). If this argument is omitted, then the **Color** property is used.



**BackColor** Optional **Variant**. The background color for the pattern. You can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (for example, red is `RGB(255, 0, 0)`). If this argument is omitted, then the **BackColor** property is used.

## Example

This example sets the interior fill of the first two series and the plot area of the first chart in ChartSpace1.

```
Sub Format_Interior_Colors()  
  
    Dim chConstants  
    Dim serSeries1  
    Dim serSeries2  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
    Set serSeries2 = ChartSpace1.Charts(0).SeriesCollection(1)  
  
    ' Set the interior fill of the first series to a one-color gradient  
    serSeries1.Interior.SetOneColorGradient chConstants.chGradientDi  
        chConstants.chGradientVariantCenter, 0.2, "Blue"  
  
    ' Set the interior fill of the second series to a preset gradient  
    serSeries2.Interior.SetPresetGradient chConstants.chGradientFrom  
        chConstants.chGradientVariantEnd, chConstants.chGradie  
  
    ' Set the interior fill of the plot area to a pattern.  
    ChartSpace1.Charts(0).PlotArea.Interior.SetPatterned chConstants  
        "Yellow", "Blue"  
  
End Sub
```



▼ [Show All](#)

# SetPresetGradient Method

-

Fills the specified **ChInterior** object with a preset gradient style.

*expression*.**SetPresetGradient**(*GradientStyle*, *gradientVariet*, *gradientPreset*)

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

**GradientStyle** Required [ChartGradientStyleEnum](#). The gradient style.

ChartGradientStyleEnum can be one of these ChartGradientStyleEnum constants.

**chGradientDiagonalDown**

**chGradientDiagonalUp**

**chGradientFromCenter**

**chGradientFromCorner**

**chGradientHorizontal**

**chGradientVertical**

*gradientVariet* Required [ChartGradientVariantEnum](#). The gradient variant.

ChartGradientVariantEnum can be one of these ChartGradientVariantEnum constants.

**chGradientVariantCenter**

**chGradientVariantEdges**

**chGradientVariantEnd**

**chGradientVariantStart**

*gradientPreset* Required [ChartPresetGradientTypeEnum](#). The gradient style used to fill the specified object.

ChartPresetGradientTypeEnum can be one of these

ChartPresetGradientTypeEnum constants.

**chGradientBrass**

**chGradientCalmWater**

**chGradientChrome**

**chGradientChromeII**

**chGradientDaybreak**

**chGradientDesert**

**chGradientEarlySunset**

**chGradientFire**

**chGradientFog**

**chGradientGold**

**chGradientGoldII**

**chGradientHorizon**

**chGradientLateSunset**

**chGradientMahogany**

**chGradientMoss**

**chGradientNightfall**

**chGradientOcean**

**chGradientParchment**

**chGradientPeacock**

**chGradientRainbow**

**chGradientRainbowII**

**chGradientSapphire**

**chGradientSilver**

**chGradientWheat**

## Example

This example sets the interior fill of the first two series and the plot area of the first chart in ChartSpace1.

```
Sub Format_Interior_Colors()  
  
    Dim chConstants  
    Dim serSeries1  
    Dim serSeries2  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
    Set serSeries2 = ChartSpace1.Charts(0).SeriesCollection(1)  
  
    ' Set the interior fill of the first series to a one-color gradient  
    serSeries1.Interior.SetOneColorGradient chConstants.chGradientDi  
        chConstants.chGradientVariantCenter, 0.2, "Blue"  
  
    ' Set the interior fill of the second series to a preset gradient  
    serSeries2.Interior.SetPresetGradient chConstants.chGradientFrom  
        chConstants.chGradientVariantEnd, chConstants.chGradi  
  
    ' Set the interior fill of the plot area to a pattern.  
    ChartSpace1.Charts(0).PlotArea.Interior.SetPatterned chConstants  
        "Yellow", "Blue"  
  
End Sub
```



# SetRootRecordset Method

Sets the root recordset for the specified **DataSourceControl** object. Use this method to change the recordset to which a data access page is bound.

*expression*.**SetRootRecordset**(*RecordsetName*, *Recordset*)

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

**RecordsetName** Required **String**. The name to use for the new recordset. If the **DataSource** control contains a recordset of this name, it will be replaced.

**Recordset** Required **Recordset** object. The ADO recordset.

## Remarks

This method supports connecting to any ADO recordset.



## Example

This example changes the root recordset used by the data source control.

```
Sub ChangeRootRecordset()  
    Dim rstCategories  
    Dim strShapeText  
    Dim strConnectionString  
  
    strShapeText = MSODSC.RootRecordsetDefs(0).ShapeText  
  
    strConnectionString = "Provider=MSDataShape.1;Persist Security In  
                           "=sqlsvr;User ID=sa;Password="";Initial Ca  
                           "Data Provider=SQLOLEDB.1"  
  
    Set rstCategories = CreateObject("ADODB.Recordset")  
  
    rstCategories.Open strShapeText, strConnectionString, 1, 3  
  
    MSODSC.SetRootRecordset "Categories", rstCategories  
  
End Sub
```



# SetSolid Method

-

Fills the specified **ChInterior** object with a solid color. Use this method to convert a gradient, textured, patterned, or background fill back to a solid fill.

*expression*.**SetSolid**(*Color*)

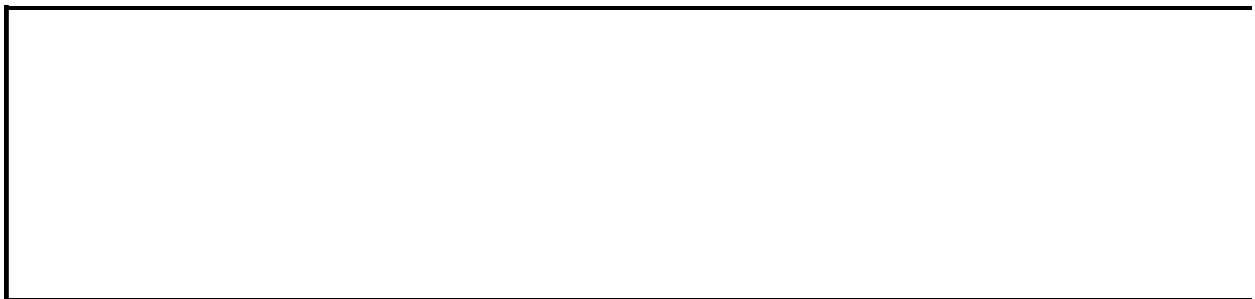
*expression* Required. An expression that returns a **ChInterior** object.

**Color** Optional **Variant**. The color for the specified **ChInterior** object. You can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (for example, red is `RGB(255, 0, 0)`).

## Example

This example sets the interior fill of the first two series and the plot area of the first chart in ChartSpace1.

```
Sub Format_Interior_Fills()  
  
    Dim chConstants  
    Dim serSeries1  
    Dim serSeries2  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
    Set serSeries2 = ChartSpace1.Charts(0).SeriesCollection(1)  
  
    ' Set the interior fill of the first series to a two-color gradi  
    serSeries1.Interior.SetTwoColorGradient chConstants.chGradientDi  
        chConstants.chGradientVariantCenter, "Blue", "Silver"  
  
    ' Set the interior fill of the second series to a solid color.  
        serSeries2.Interior.SetSolid "Purple"  
  
    ' Set the interior fill of the plot area to a preset texture.  
    ChartSpace1.Charts(0).PlotArea.Interior.SetTextured _  
        chConstants.chTextureParchment, chConstants.chTile  
  
End Sub
```



# SetSpreadsheetData Method

-

Binds the specified chart or chartspace to a range of cells on on a worksheet in the **Spreadsheet** Component. The chart must already be bound to a range of cells in the **Spreadsheet** Component before you call this method.

*expression*.**SetSpreadsheetData**(*DataReference*, *SeriesByRows*)

*expression* Required. An expression that returns a **ChartSpace** or **ChChart** object.

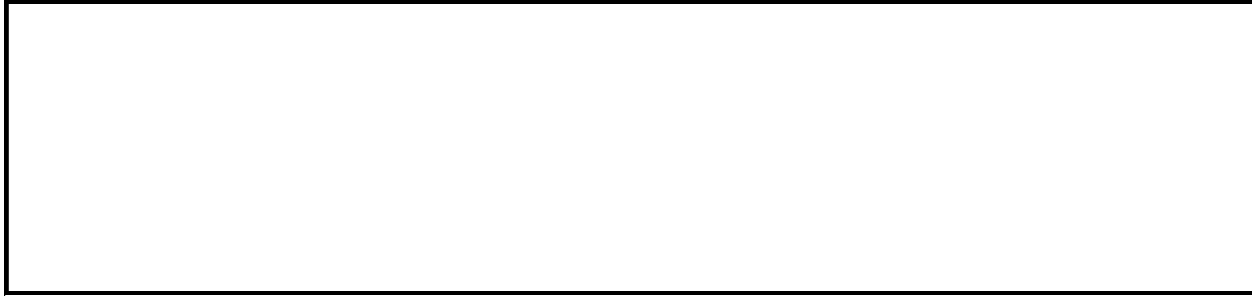
**DataReference** Required **String**. A reference to the range of cells to bind to. This can be in the form of a cell reference (A1:D5), or a defined name.

**SeriesByRows** Optional **Boolean**. Specifies whether or not each row represents a data series. Set this property to **False** if each column represents a data series.

## Example

This example binds ChartSpace1 to cells A1:F25 in the first sheet of Spreadsheet1.

```
ChartSpace1.SetSpreadsheetData "Sheet1!A1:F25", False
```



▼ [Show All](#)

# SetTextured Method

-

Fills the specified **ChInterior** object with an image or a preset texture.

*expression*.**SetTextured**(*textureFile*, *TextureFormat*, *stackUnit*, *TexturePlacement*)

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

**textureFile** Required **Variant**. The image used to fill the interior of the object. You can specify a URL that points to an image file or a [ChartPresetTextureEnum](#) constant.

ChartPresetTextureEnum can be one of these ChartPresetTextureEnum constants.

**chTextureBlueTissuePaper**

**chTextureBouquet**

**chTextureBrownMarble**

**chTextureCanvas**

**chTextureCork**

**chTextureDenim**

**chTextureFishFossil**

**chTextureGranite**

**chTextureGreenMarble**

**chTextureMediumWood**

**chTextureNewsprint**

**chTextureOak**

**chTexturePaperBag**

**chTexturePapyrus**

**chTextureParchment**

**chTexturePinkTissuePaper**

**chTexturePurpleMesh**

**chTextureRecycledPaper**  
**chTextureSand**  
**chTextureStationery**  
**chTextureWalnut**  
**chTextureWaterDroplets**  
**chTextureWhiteMarble**  
**chTextureWovenMat**

**TextureFormat** Optional [ChartTextureFormatEnum](#). Determines how the picture is displayed within the specified **ChInterior** object.

ChartTextureFormatEnum can be one of these ChartTextureFormatEnum constants.

**chStack**  
**chStackScale**  
**chStretch**  
**chStretchPlot**  
**chTile** *default*

**stackUnit** Optional **Double**. Specifies how the picture is stacked and scaled when the **TextureFormat** argument is set to **chStackScale**.

**TexturePlacement** Optional [ChartTexturePlacementEnum](#). Affects where the picture is displayed within the specified **ChInterior** object. This setting only affects 3-D charts.

ChartTexturePlacementEnum can be one of these ChartTexturePlacementEnum constants.

**chAllFaces** *default*  
**chEnd**  
**chEndSides**  
**chFront**  
**chFrontEnd**  
**chFrontSides**  
**chProjectFront**

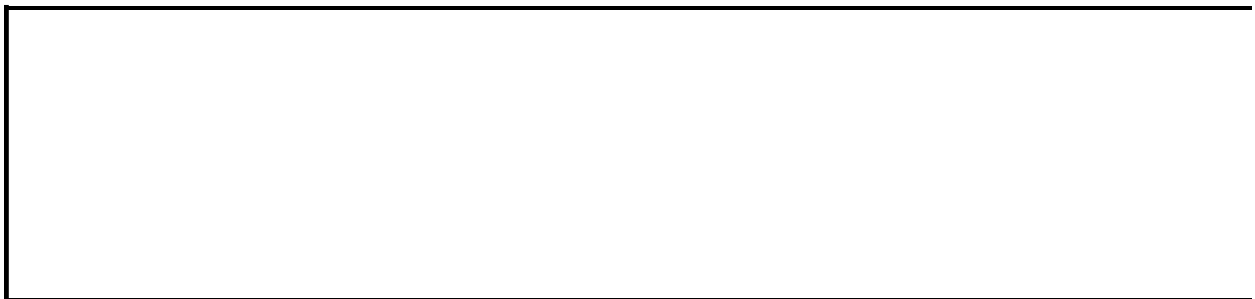


**chSides**

## Example

This example sets the interior fill of the first two series and the plot area of the first chart in ChartSpace1.

```
Sub Format_Interior_Fills()  
  
    Dim chConstants  
    Dim serSeries1  
    Dim serSeries2  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
    Set serSeries2 = ChartSpace1.Charts(0).SeriesCollection(1)  
  
    ' Set the interior fill of the first series to a two-color gradi  
    serSeries1.Interior.SetTwoColorGradient chConstants.chGradientDi  
        chConstants.chGradientVariantCenter, "Blue", "Silver"  
  
    ' Set the interior fill of the second series to a solid color.  
    serSeries2.Interior.SetSolid "Purple"  
  
    ' Set the interior fill of the plot area to a preset texture.  
    ChartSpace1.Charts(0).PlotArea.Interior.SetTextured _  
        chConstants.chTextureParchment, chConstants.chTile  
  
End Sub
```



▼ [Show All](#)

# SetTwoColorGradient Method

-

Fills the specified **ChInterior** object with a two-color gradient.

*expression*.**SetTwoColorGradient**(*GradientStyle*, *GradientVariant*, *Color*, *BackColor*)

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

*GradientStyle* Required [ChartGradientStyleEnum](#). The gradient style.

ChartGradientStyleEnum can be one of these ChartGradientStyleEnum constants.

**chGradientDiagonalDown**

**chGradientDiagonalUp**

**chGradientFromCenter**

**chGradientFromCorner**

**chGradientHorizontal**

**chGradientVertical**

*GradientVariant* Required [ChartGradientVariantEnum](#). The gradient variant. If *GradientStyle* is **chGradientFromCenter**, the *GradientVariant* argument can only be **chGradientVariantStart** or **chGradientVariantEnd**.

ChartGradientVariantEnum can be one of these ChartGradientVariantEnum constants.

**chGradientVariantCenter**

**chGradientVariantEdges**

**chGradientVariantEnd**

**chGradientVariantStart**

*Color* Optional **Variant**. The foreground color of the gradient. You can use either a **Long** value representing a red-green-blue color value or a **String** value

naming a valid HTML color value. In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (for example, red is `RGB(255, 0, 0)`). If omitted, the **Color** property is used to determine the foreground color.

**BackColor** Optional **Variant**. The background color of the gradient. You can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (for example, red is `RGB(255, 0, 0)`). If omitted, the **BackColor** property is used to determine the foreground color.

## Example

This example sets the interior fill of the first two series and the plot area of the first chart in ChartSpace1.

```
Sub Format_Interior_Fills()  
  
    Dim chConstants  
    Dim serSeries1  
    Dim serSeries2  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
    Set serSeries2 = ChartSpace1.Charts(0).SeriesCollection(1)  
  
    ' Set the interior fill of the first series to a two-color gradi  
    serSeries1.Interior.SetTwoColorGradient chConstants.chGradientDi  
        chConstants.chGradientVariantCenter, "Blue", "Silver"  
  
    ' Set the interior fill of the second series to a solid color.  
    serSeries2.Interior.SetSolid "Purple"  
  
    ' Set the interior fill of the plot area to a preset texture.  
    ChartSpace1.Charts(0).PlotArea.Interior.SetTextured _  
        chConstants.chTextureParchment, chConstants.chTile  
  
End Sub
```



# Show Method

-

Scrolls the spreadsheet window to move the specified range into view.

*expression*.**Show**

*expression* An expression that returns a [Range](#) object.

## Remarks

If you specify a range that is not on the active worksheet, this method will not activate that worksheet. However, the specified range will be visible when you activate that worksheet. For example, if Sheet1 is currently active and you use this method with cell A500 on Sheet2, Sheet1 remains active. Cell A500 will be active if you immediately activate Sheet2, either manually or by using the *Activate* method of the **Worksheet** object.



## Example

This example scrolls the spreadsheet until cell P75 is visible.

```
Spreadsheet1.ActiveSheet.Range("p75").Show
```



# ShowAbout Method

-  
Displays the **About Microsoft Office Web Components** dialog box.

*expression*.**ShowAbout**

*expression* Required. An expression that returns a **Spreadsheet** or **PivotTable** object.



# ShowAllData Method

-  
Makes all filtered rows visible on the specified worksheet and sets all filters to Show All.

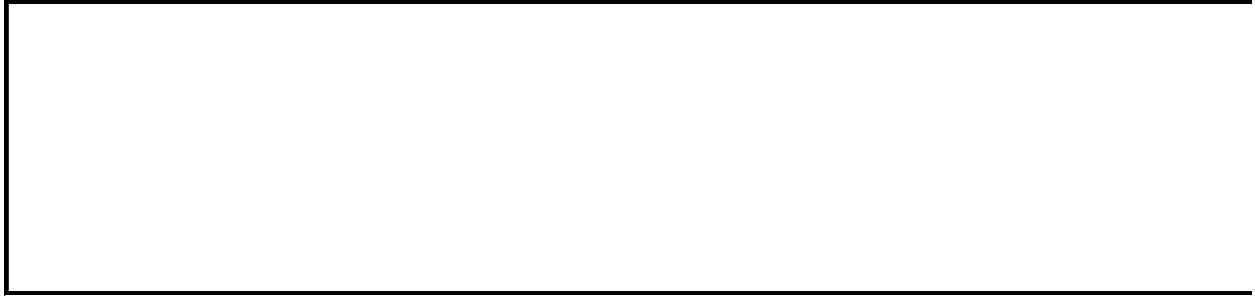
*expression*.**ShowAllData**

*expression* An expression that returns a [Worksheet](#) object.

## Example

This example makes all filtered rows visible on the active worksheet and sets all filters to Show All.

`Spreadsheet1.ActiveSheet.ShowAllData`



# ShowContextMenu Method

-

Displays a customized context menu at the specified screen coordinates.

*expression*.**ShowContextMenu**(*x*, *y*, *Menu*)

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

*x* Required **Long**. Represents the x-coordinate where the context menu is to appear.

*y* Required **Long**. Represents the y-coordinate where the context menu is to appear.

*Menu* Required **Variant**. The array that contains the menu items to display.

## Example

This example displays a context menu.

```
Sub ShowMenu()  
  
    Dim cmContextMenu(4)  
    Dim cmClearSubMenu(2)  
  
    cmClearSubMenu(0) = Array("&All", "ClearAll")  
    cmClearSubMenu(1) = Array("&Formats", "ClearFormats")  
    cmClearSubMenu(2) = Array("&Values", "ClearValues")  
  
    cmContextMenu(0) = Array("Cu&t", "owc2")  
    cmContextMenu(1) = Array("&Copy", "owc3")  
    cmContextMenu(2) = Array("&Paste", "owc4")  
    cmContextMenu(3) = Empty  
    cmContextMenu(4) = Array("Clea&r", cmClearSubMenu)  
  
    Spreadsheet1.ShowContextMenu 10, 40, cmContextMenu  
  
End Sub
```



# ShowDetails Method

-

Expands the details cells for the specified object. If the specified object is a **PivotData** object, all detail cells are expanded. If the specified object is a **PivotRowMember** object, then all details cells in that row are expanded. If the specified object is a **PivotColumnMember** object, then all details cells in that column are expanded.

*expression*.**ShowDetails**

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

## Example

This example expands all of the detail cells in PivotTable1.

PivotTable1.ActiveData.**ShowDetails**





▼ [Show All](#)

# ShowHelp Method

▸ [ShowHelp method as it applies to the \*\*ChartSpace\*\* object.](#)

Displays the main Help topic for the **Chart** control.

*expression*.**ShowHelp**(*iTopic*)

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

*iTopic* Required **Long**. Specify 0 to show the main Help topic for the **Spreadsheet** control.

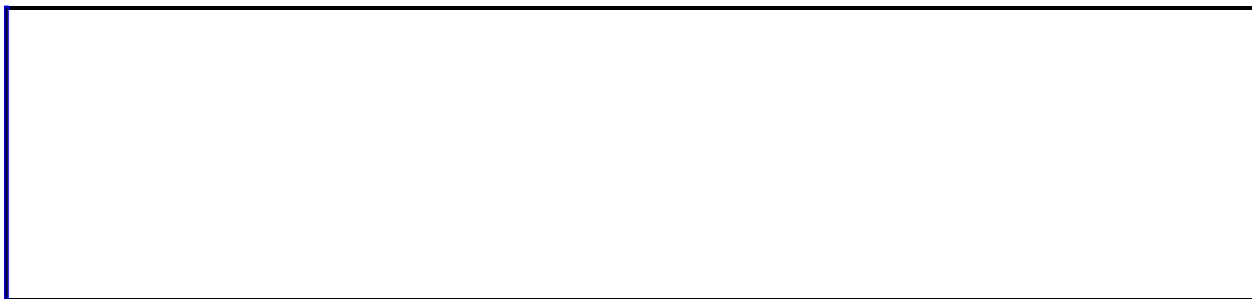
▸ [ShowHelp method as it applies to the \*\*PivotTable\*\* and \*\*Spreadsheet\*\* objects.](#)

Displays the main Help topic for the **PivotTable** or **Spreadsheet** control.

*expression*.**ShowHelp**(*Topic*)

*expression* Required. An expression that returns one of the above objects.

*Topic* Required **Long**. Specify 0 to show the main Help topic for the control.



# SmallScroll Method

-

Scrolls the contents of the window by rows or columns.

*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

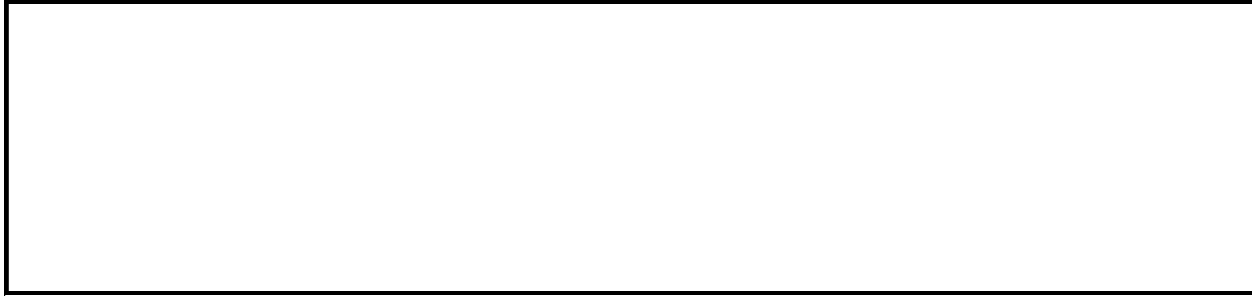
If ***Down*** and ***Up*** or ***ToLeft*** and ***ToRight*** are both specified, the contents of the window are scrolled by the difference of the arguments. For example, if ***Down*** is 3 and ***Up*** is 6, the the contents are scrolled up three rows.

Any of these arguments can be a negative number.

## Example

This example scrolls the contents of the active window of Spreadsheet1 down three rows.

Spreadsheet1.ActiveWindow.**SmallScroll** 3



▼ [Show All](#)

# Sort Method

-  
Sorts a range or, if the specified range contains only one cell, the current region  
*expression*.**Sort**(*ColumnKey*, *Order*, *Header*)

*expression* An expression that returns a [Range](#) object.

**ColumnKey** Optional **Long**. The number of the first sort column. The default value is 1.

**Order** Optional [XLSortOrder](#). The sort order.

XLSortOrder can be one of these XLSortOrder constants.

**xlAscending** *default*

**xlDescending**

**Header** Optional [XLYesNoGuess](#). Determines whether the first row contains headers.

XLYesNoGuess can be one of these XLYesNoGuess constants.

**xlGuess** Have the method determine whether the first row contains headers.

**xlNo** *default* The first row does not contain headers.

**xlYes** The first row contains headers.

## Example

This example sorts the range A1:F10 in descending order based on column B.

```
Sub SortData()  
    Dim ssConstants  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Sort Range A1:F10 on column B in descending order and specify  
that  
    ' row 1 contains headings.  
    Spreadsheet1.ActiveSheetRange("A1:F10").Sort 2, _  
        ssConstants.xlDescending, ssConstants.xlYes  
End Sub
```





# SortAscending Method

-  
Sorts a field on a data access page in ascending order.

*expression*.**SortAscending**

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

## Remarks

This method relies upon the current selection on the data access page to determine the field to sort by. Therefore, you must set the focus to the field to sort by, when the procedure containing this method is invoked by a control on the data access page, such as a command button.

## Example

This example sorts the ProductName field in ascending order.

```
Sub SortProductNameAscending()  
    ' Set focus to the control for the ProductName field.  
    MSODSC.Datapages(0).FirstSection.HTMLContainer.Children("ProductN  
  
    ' Sort the field in descending order.  
    MSODSC.DataPages(0).SortAscending  
  
End Sub
```



# SortDescending Method

-  
Sorts a field on a data access page in descending order.

*expression*.**SortDescending**

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

## Remarks

This method relies upon the current selection on the data access page to determine the field to sort by. Therefore, you must set the focus to the field to sort by, when the procedure containing this method is invoked by a control on the data access page, such as a command button.

## Example

This example sorts the ProductName field in descending order.

```
Sub SortProductNameDescending()  
    ' Set focus to the control for the ProductName field.  
    MSODSC.Datapages(0).FirstSection.HTMLContainer.Children("ProductN  
  
    ' Sort the field in descending order.  
    MSODSC.DataPages(0).SortDescending  
  
End Sub
```



▼ [Show All](#)

# StartEdit Method

-

Places the active detail cell into edit mode.

*expression*.**StartEdit**(*InitialValue*, *ArrowMode*, *CaretPosition*)

*expression* Required. An expression that returns one of the a **PivotTable** object.

**InitialValue** Optional **Variant**. Specifies the initial value to use when editing the cell. The current value is used if you do not specify a value for this argument.

**ArrowMode** Optional [PivotArrowModeEnum](#). Specifies how the left and right arrows function while the user is in edit mode.

PivotArrowModeEnum can be one of these PivotArrowModeEnum constants.

**plArrowModeAccept** *default* Accept the edit and move to the next cell.

**plArrowModeEdit** Move the insertion point left or right one position within the cell.

**CaretPosition** Optional [PivotCaretPositionEnum](#). Specifies the position of the insertion point within the cell.

PivotCaretPositionEnum can be one of these PivotCaretPositionEnum constants.

**plCaretPositionAtEnd** *default*

**plCaretPositionAtMouse**



## Remarks

This method will result in a run-time error if the current selection is not a detail cell, or if the current data is not editable.

# TextHeight Method

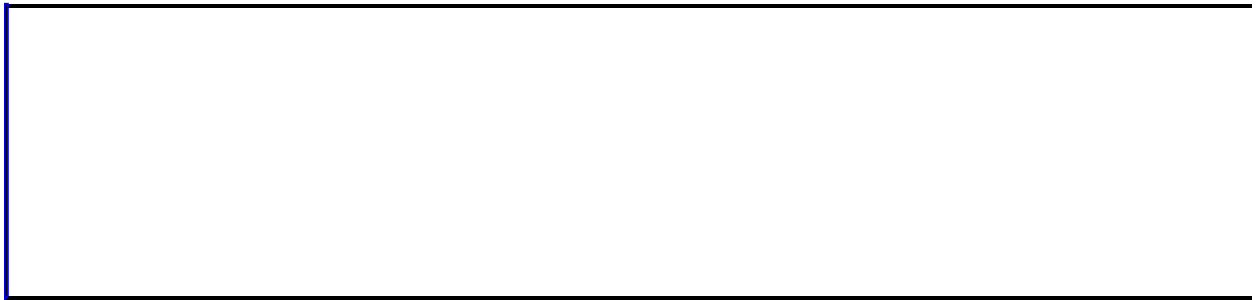
-

Calculates and returns a **Variant** that represents the width of the specified text in pixels, based on the current font setting.

*expression*.**TextHeight**(*Text*)

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

*Text* Required **String**. The text whose size you want to calculate.



# TextWidth Method

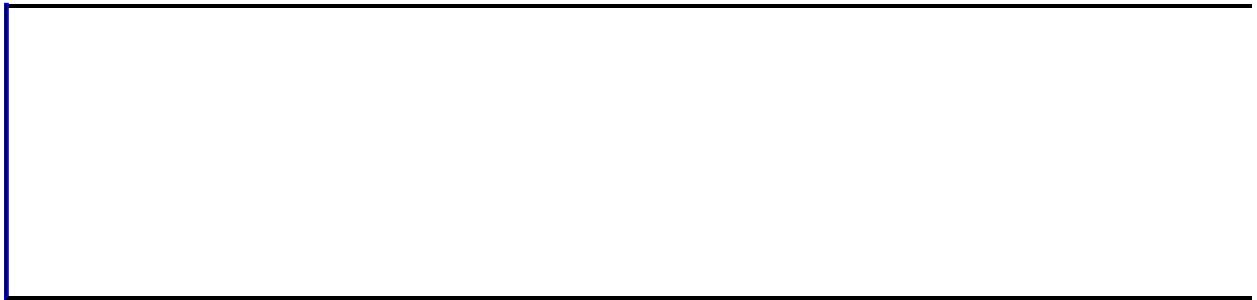
-

Calculates and returns a **Variant** that represents the width of the specified text in pixels, based on the current font setting.

*expression*.**TextWidth**(*Text*)

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

*Text* Required **String**. The text whose size you want to calculate.



# ToggleFilter Method

-

Toggles the state of the current filter that has been applied to the data access page. If the filter is active, calling this method deactivates the filter. Calling this method a second time reapplies the filter.

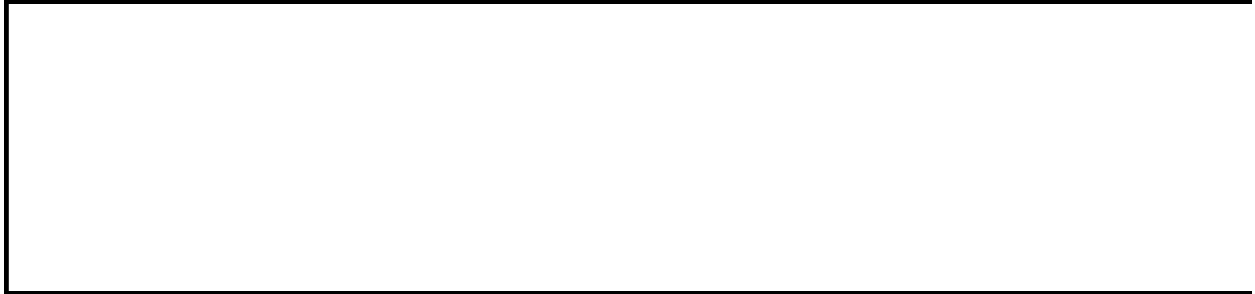
*expression*.**ToggleFilter**

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

## Example

This example toggles the filter on the first data access page in the data source control named MSODSC.

```
MSODSC.DataPages(0).ToggleFilter
```



# Undo Method

-

For the [ChartSpace](#) and [Spreadsheet](#) objects, undoes the last single action or the last macro block surrounded by [BeginUndo](#) and [EndUndo](#) method calls.

For the [DataPage](#) object, restores the data access page to the condition before the recordset was edited if the edits have not been saved.

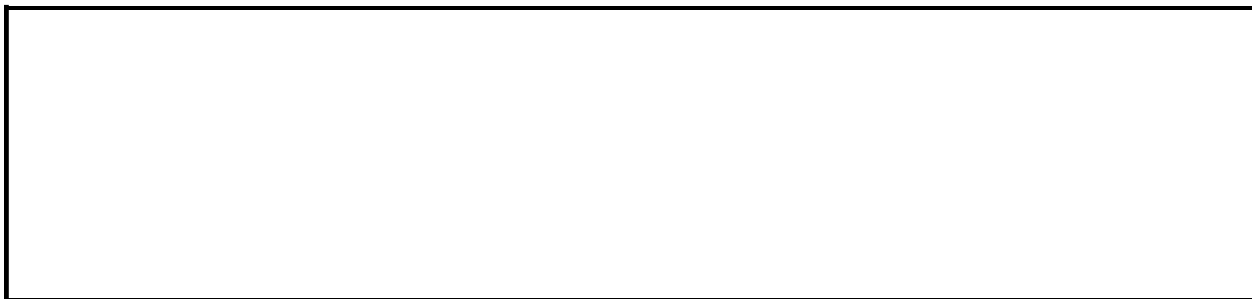
*expression*.**Undo**

*expression* An expression that returns a **ChartSpace**, **DataPage**, or **Spreadsheet** object.

## Example

This example undoes the last action or displays a message box if this action cannot be undone.

```
If Spreadsheet1.CanUndo Then
    Spreadsheet1.Undo
Else
    MsgBox "can't undo last action"
End If
```



# Ungroup Method

Moves the specified series into a new layer.

*expression*.**Ungroup**(*UseNewScaling*)

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

***UseNewScaling*** Optional **Boolean**. Set this argument to **True** to specify that the series uses a different scaling than the previous layer. The default value is **False**.



## Remarks

When you move a series into a new layer, you can assign the series to an axis that is based on a different value scale.

## Example

This example creates a combination chart based on literal data. The first data series is plotted as a line. The second data series is plotted as columns, and on a it's own value axis.

```
Sub Window_Onload()  
  
    Dim asSeriesNames(1)  
    Dim asCategories(3)  
    Dim aiSeries1(3)  
    Dim alSeries2(3)  
    Dim chConstants  
    Dim chtNewChart  
    Dim serUnitSales  
    Dim serDispInc  
    Dim axIncomeAxis  
  
    asSeriesNames(0) = "UnitSales"  
    asSeriesNames(1) = "Disposable Income"  
  
    asCategories(0) = "Item 1"  
    asCategories(1) = "Item 2"  
    asCategories(2) = "Item 3"  
    asCategories(3) = "Item 4"  
  
    aiSeries1(0) = 75  
    aiSeries1(1) = 84  
    aiSeries1(2) = 30  
    aiSeries1(3) = 94  
  
    alSeries2(0) = 14522  
    alSeries2(1) = 17321  
    alSeries2(2) = 9424  
    alSeries2(3) = 41782  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Enable the display of the legend.  
    ChartSpace1.HasChartSpaceLegend = True  
  
    ' Add a new chart to Chartspace1.  
    Set chtNewChart = ChartSpace1.Charts.Add  
  
    ' Specify that the chart is a column chart.  
    chtNewChart.Type = chConstants.chChartTypeLineMarkers
```

```

' Bind the chart to the arrays.
chtNewChart.SetData chConstants.chDimSeriesNames, chConstants.ch
chtNewChart.SetData chConstants.chDimCategories, chConstants.chD

Set serUnitSales = chtNewChart.SeriesCollection(0)

serUnitSales.SetData chConstants.chDimValues, chConstants.chData

Set serDispInc = chtNewChart.SeriesCollection(1)

serDispInc.SetData chConstants.chDimValues, chConstants.chDataLi

' Ungroup the series.
serDispInc.Ungroup True

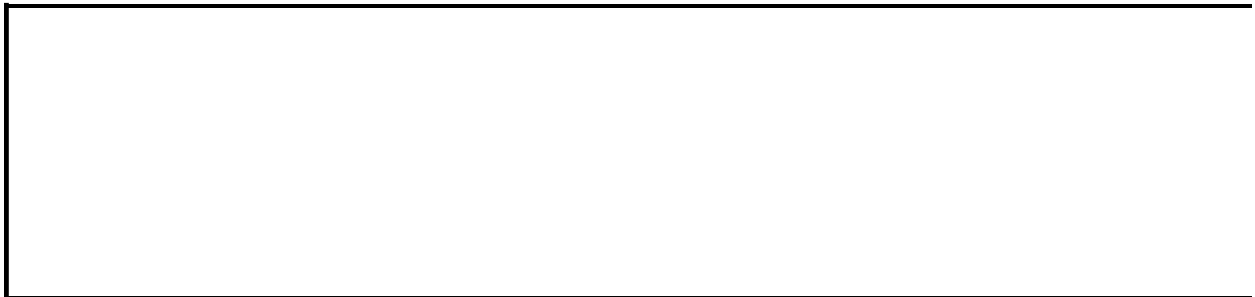
' Add a new value axis to the chart based on the values in the s
Set axIncomeAxis = chtNewChart.Axes.Add(serDispInc.Scalings(chCo

' Place the axis on the right side of the chart.
axIncomeAxis.Position = chConstants.chAxisPositionRight

' Display the series as columns.
serDispInc.Type = chConstants.chChartTypeColumnClustered

```

End Sub



# UnMerge Method

-

Separates the specified merged area into individual cells. When you separate a merged area, the value in the merged area is placed in the cell in the upper-left corner of the area. All other cells are cleared.

*expression*.**UnMerge**

*expression* An expression that returns a [Range](#) object.

## Example

This example separates the merged area containing cell A1.

```
Spreadsheet1.ActiveSheet.Range("A1").UnMerge
```



# Unprotect Method

-

Removes protection from a worksheet or workbook. This method has no effect if the worksheet or workbook isn't protected. This method is equivalent to setting the **Enabled** property of the **Protection** object to **False**.

*expression*.**Unprotect**(*Password*)

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

***Password*** This argument is not supported in this version of the Microsoft Office Spreadsheet Component.

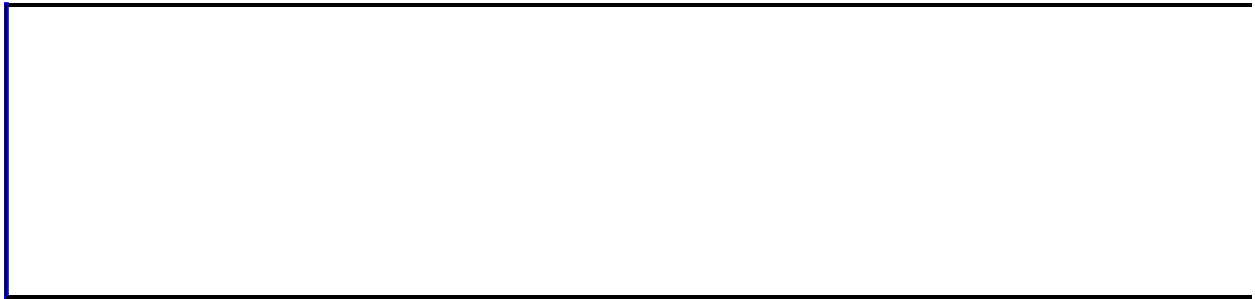


# UpdatePropertyToolbox Method

-  
Updates the Commands and Options window from the currently selected object.

*expression*.**UpdatePropertyToolbox**

*expression* An expression that returns a [Spreadsheet](#) object.



▼ [Show All](#)



# ValueToPoint Method

► [ValueToPoint method as it applies to the \*\*ChAxis\*\* object.](#)

Returns a **Coordinate** object that contains information about pixel coordinates of a data point on the specified axis.

*expression*.**ValueToPoint**(*Value*)

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

**Value** Required **Variant**. The value that you want to return the pixel coordinates for.

► [ValueToPoint method as it applies to the \*\*ChSeries\*\* object.](#)

Returns a **Coordinate** object that contains information about pixel coordinates for the specified data point.

*expression*.**ValueToPoint**(*xvalue*, *yvalue*, *zvalue*)

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

**xvalue** Required **Variant**. The value or category label on the category axis.

**yvalue** Required **Variant**. The value on the value axis.

**zvalue** Optional **Variant**. The value or series label on the series axis on 3D charts.

## Remarks

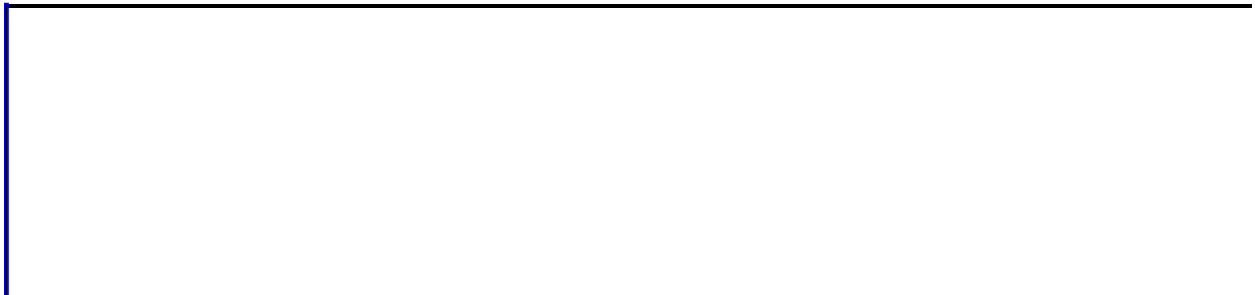
Use the **x** and **y** properties of the returned **Coordinate** object to return the X and Y-coordinates for the specified data point.

## Example

► [As it applies to the \*\*ChSeries\*\* object.](#)

This example changes the title of the first chart in Chartspace1 to the pixel coordinates of a data point in the first series of the chart.

```
Sub GetPixelCoordinates()  
  
    Dim chChart1  
    Dim lXPos  
    Dim lYPos  
    Dim coPointCoordinates  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set chChart1 = ChartSpace1.Charts(0)  
  
    ' Enable the title for the chart.  
    chChart1.HasTitle = True  
  
    ' Set a Coordinate object to the coordinates of a data point.  
    Set coPointCoordinates = chChart1.SeriesCollection(0).ValueToPoi  
  
    ' Set a variable to the X-coordinate.  
    lXPos = coPointCoordinates.x  
  
    ' Set a variable to the Y-coordinate.  
    lYPos = coPointCoordinates.y  
  
    ' Set the chart's title to the pixel coordinates of the specifie  
    ' data point.  
    chChart1.Title.Caption = "X(" & lXPos & ") Y(" & lYPos & ")"  
  
End Sub
```



# ActiveCell Property

-  
Returns a [Range](#) object that represents the active cell. Read-only.

*expression*.**ActiveCell**

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

## Remarks

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.

## Example

This example sets the number format in the active cell on the worksheet.

```
Spreadsheet1.ActiveCell.NumberFormat = "0.##"
```



# ActiveData Property

-

Returns a [PivotData](#) object that represents the data in the active PivotTable list.

*expression*.**ActiveData**

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

---

---

# ActiveObject Property

-

Returns or sets an **Object** that represents the selected cell in the detail area of the PivotTable list. Use the **Value** property of the returned object to change the value of the selected cell.

*expression*.**ActiveObject**

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

## Remarks

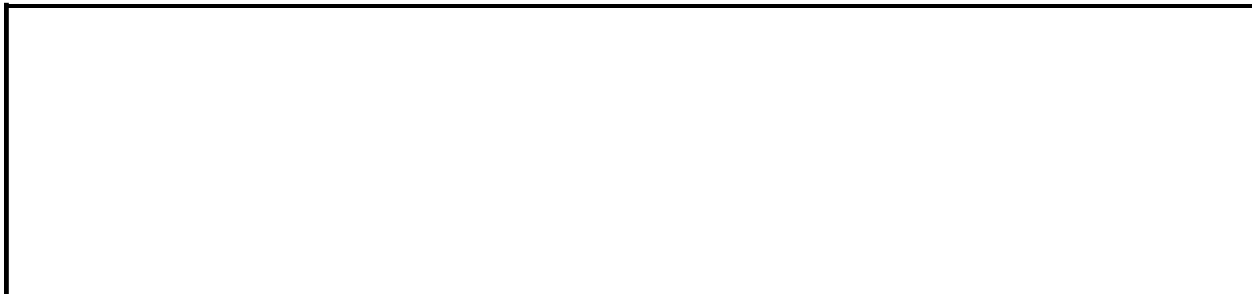
This property returns **Nothing** if no detail cells are selected.



## Example

This example enables the editing of detail records in PivotTable1, then places the current date in the selected detail cell. The editing of detail records is disallowed once the date has been inserted into the selected cell.

```
Sub EditSelectedCell()  
  
    Dim objActiveCell  
  
    ' Allow editing of detail records.  
    PivotTable1.ActiveView.AllowEdits = True  
  
    ' Set a variable to the currently selected cell.  
    Set objActiveCell = PivotTable1.ActiveObject  
  
    ' Check whether a detail cell is selected.  
    If Not objActiveCell Is Nothing Then  
  
        ' Set the value of the detail cell  
        ' to the current date.  
        ' Note: This will result in a run-time  
        ' error if the data type of the selected  
        ' cell does not support date values.  
        objActiveCell.Value = Date  
  
    End If  
  
    ' Disallow editing of detail records.  
    PivotTable1.ActiveView.AllowEdits = False  
  
End Sub
```



# ActivePane Property

-

Returns a [Pane](#) object that represents the active spreadsheet pane. Read-only.

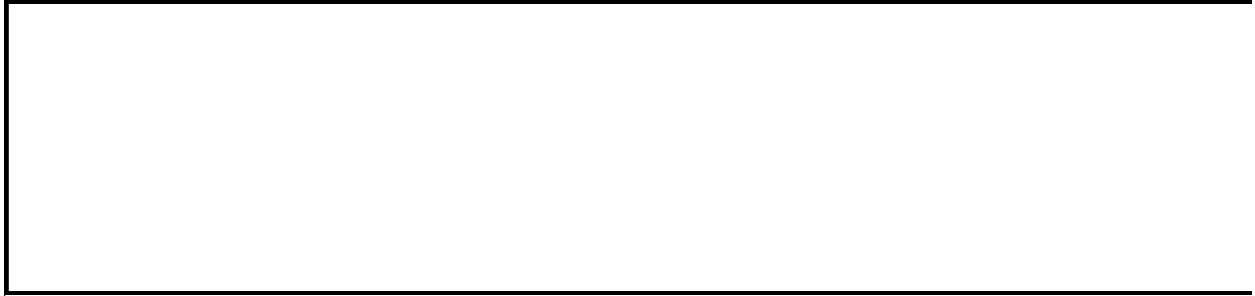
*expression*.**ActivePane**

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

## Example

This example sets the value of the cell in the upper-left corner of the visible range in the active pane on the spreadsheet.

```
Spreadsheet1.ActivePane.VisibleRange.Cells(1, 1).Value = "top left"
```



# ActiveSheet Property

-  
Returns a read-only [Worksheet](#) object that represents the active worksheet.

*expression*.**ActiveSheet**

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

## Example

This example changes the name of the active worksheet.

```
Spreadsheet1.ActiveSheet.Name = "Budget Sheet"
```



# ActiveView Property

-

Returns a [PivotView](#) object that represents the layout of the active PivotTable list. Setting a variable to the active view of the PivotTable provides you with a convenient method to make changes to the active view. Read-only.

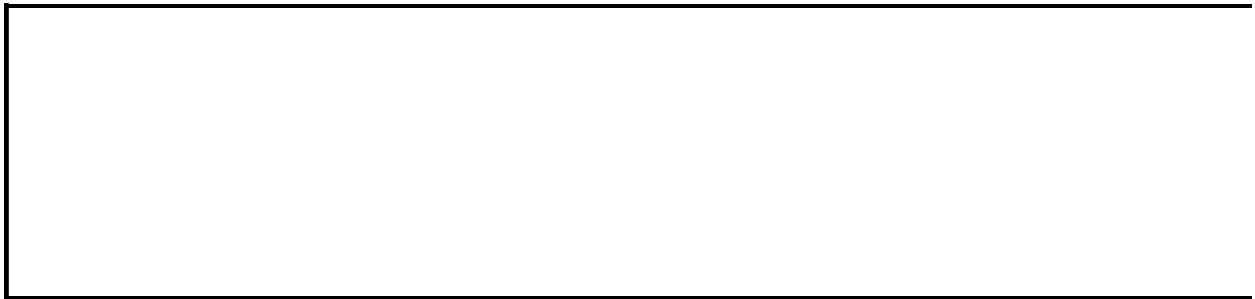
*expression*.**Case**

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

## Example

This example inserts two fieldsets in the active view of PivotTable1.

```
Sub Insert_Fieldsets()  
    Dim vwView  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' Add the Store Type field to the column axis.  
    vwView.ColumnAxis.InsertFieldSet vwView.FieldSets("Store Type")  
  
    ' Add the Promotions field to the row axis.  
    vwView.RowAxis.InsertFieldSet vwView.FieldSets("Promotions")  
End Sub
```



# ActiveWindow Property

-  
Returns a **Window** object that represents the current window.

*expression*.**ActiveWindow**

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



## Example

This example hides the row and column headings in the active window of Spreadsheet1.

```
Sub HideHeadings()  
    Spreadsheet1.ActiveWindow.DisplayColumnHeadings = False  
    Spreadsheet1.ActiveWindow.DisplayRowHeadings = False  
End Sub
```



# ActiveWorkbook Property

Returns a **Workbook** object that represents the open workbook.

*expression*.**ActiveWorkbook**

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

## Example

This example protects the structure of the open workbook in Spreadsheet1.

```
Speadsheet1.ActiveWorkbook.Protect , True
```

▼ [Show All](#)

# Address Property

## ▸ [Address property as it applies to the \*\*Hyperlink\*\* and \*\*PivotHyperlink\*\* objects.](#)

Returns or sets a **String** that represents the address of the target document.  
Read/write.

*expression*.**Address**

*expression* Required. An expression that returns a **Hyperlink** or **PivotHyperlink** object.

## ▸ [Address property as it applies to the \*\*Range\*\* object.](#)

Returns a **String** that represents the address of the specified range (for example, A1). Read-only.

*expression*.**Address**(*RowAbsolute*, *ColumnAbsolute*, *ReferenceStyle*, *External*, *RelativeTo*)

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

**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 [xlReferenceStyle](#). The reference style.

ReferenceStyle can be one of the following XlReferenceStyle constants.

**xlA1**

**xlR1C1**

***External*** Optional **Variant**. **True** to return an external reference. An external reference includes the name of the worksheet. **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.

## Example

► [As it applies to the \*\*Hyperlink\*\* object.](#)

The following example inserts a hyperlink into cell C10 of Sheet1 in Spreadsheet1.

```
Sub Insert_Hyperlink()  
    Dim rngNewHyperlink  
  
    Set rngNewHyperlink = Spreadsheet1.Worksheets("sheet1").Range("c  
        rngNewHyperlink.Value = "Click to read about the latest Office u  
        rngNewHyperlink.Hyperlink.Address = "http://officeupdate.microso  
End Sub
```

► [As it applies to the \*\*Range\*\* object.](#)

This example returns the address of the visible range.

```
MsgBox Spreadsheet1.ActiveSheet.VisibleRange.Address
```



# Aggregates Property

-  
Returns the [PivotAggregates](#) collection for the specified cell.

*expression*.**Aggregates**

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



## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# AllGroupingDefs Property

-

Returns the [AllGroupingDefs](#) collection for the data source control. Read-only.

---

*expression*.**AllGroupingDefs**

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

▼ [Show All](#)

# AllIncludeExclude Property

-

Returns or sets a [PivotFieldSetAllIncludeExcludeEnum](#) that represents the inclusion state of all members that are not listed in the **IncludedMembers** or **ExcludedMembers** lists for the specified field set. Read/write.

PivotFieldSetAllIncludeExcludeEnum can be one of these  
PivotFieldSetAllIncludeExcludeEnum constants.

**plAllDefault** *Default* All members are included when the field set is on the row or column axis of a PivotTable.

**plAllExclude**

**plAllInclude**

*expression*.**AllIncludeExclude**

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

## Example

This example sets the included and excluded members of the Store State and Store City fields in PivotTable1. Members not listed in the include and exclude lists are excluded.

```
Sub MemberFiltering()  
  
    Dim fldStoreCity  
    Dim fldStoreState  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Store State field.  
    Set fldStoreState = ptView.FieldSets("Store").Fields("Store State")  
  
    ' Set a variable to the Store City field.  
    Set fldStoreCity = ptView.FieldSets("Store").Fields("Store City")  
  
    ' Exclude California and Washington from the Store State field.  
    fldStoreState.ExcludedMembers = Array("CA", "WA")  
  
    ' Include members of the Store City field. Note that the cities  
    ' in states that have been excluded by the previous line. Since  
    ' Store State is a parent to Store City, then the excluded state  
    ' are displayed in the PivotTable.  
    fldStoreCity.IncludedMembers = Array("Los Angeles", "San Diego",  
                                         "Seattle", "Spokane")  
  
    ' Exclude all members that are not in the list for the IncludedMembers  
    ' ExcludedMembers properties.  
    ptView.FieldSets("Store").AllIncludeExclude = ptConstants.plAllMembers  
  
End Sub
```

--

# AllMember Property

-

Returns a **PivotMember** object that represents the top member in the specified field set.

*expression*.**AllMember**

*expression* Required. An expression that returns **PivotFieldSet** object.



# AllowAdditions Property

**GroupLevel object:** Returns or sets a **Boolean** that represents whether the user can add records to the specified group. Set this property to **False** to prevent users from adding records to a group. The default value is **True**. Read/write.

**PivotView object:** Returns or sets a **Boolean** that represents whether the user can add a new record to the detail area of a PivotTable list. Set this property to **True** to allow the user to insert new records. The default value is **False**. Read/write.

*expression*.**AllowAdditions**

*expression* Required. An expression that returns a **GroupLevel** or **PivotView** object.



## Remarks

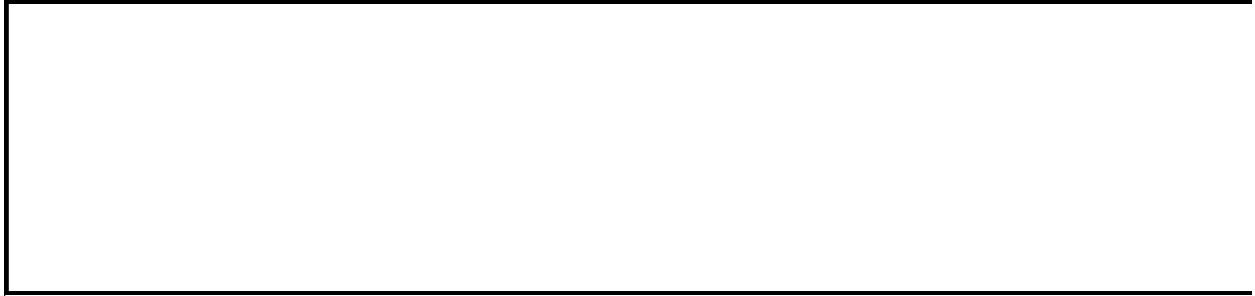
When this property is set to **True**, an asterisk (\*) will be displayed in a blank row in the detail area. Any new records added to the detail area of a PivotTable list are also added to the source database.

You cannot add records to the detail area of the PivotTable list if the source recordset does not allow insertions.

## Example

This example allows the user to add new records to the detail area of PivotTable1.

```
PivotTable1.ActiveView.AllowAdditions = True
```



# AllowCustomOrdering Property

-

Returns or sets whether the user can reorder row axis or column axis members. Set this property to **False** to prevent users from row axis or column axis member reordering. The default value is **True**. Read/write **Boolean**.

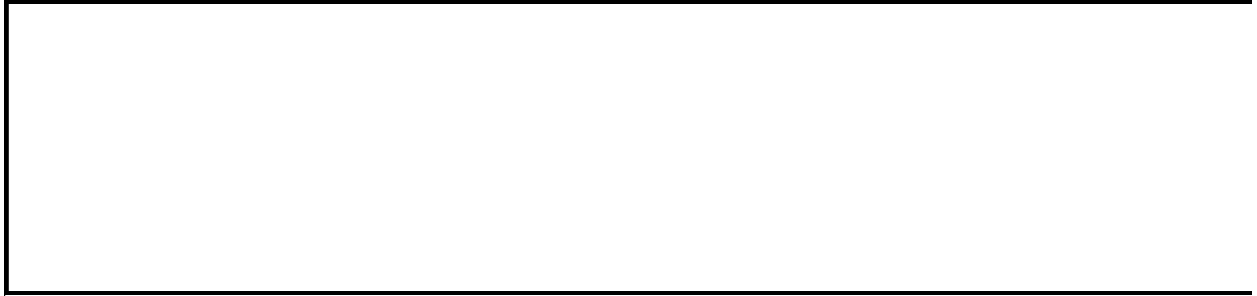
*expression*.**AllowCustomOrdering**

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

## Example

This example prevents users from reordering members of the row axis or column axis in PivotTable1.

```
PivotTable1.AllowCustomOrdering = False
```



# AllowDeletingColumns Property

-  
Specifies whether a worksheet column can be deleted when the worksheet has been protected. The default value is **False**, but this property has no effect if the **Enabled** property of the **Protection** object is set to **False**. Read/write **Boolean**.

*expression*.**AllowDeletingColumns**

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

## Example

This example locks all cells on Sheet1, enables the insertion and deletion of columns, and then protects Sheet1.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    'Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to delete columns while Sheet1 is protected.  
    ptProtSheet1.AllowDeletingColumns = True  
  
    ' Allows user to insert columns while Sheet1 is protected.  
    ptProtSheet1.AllowInsertingColumns = True  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
End Sub
```



# AllowDeletingRows Property

-  
Specifies whether a worksheet row can be deleted when the worksheet has been protected. The default value is **False**, but this property has no effect if the [Enabled](#) property of the **Protection** object is set to **False**. Read/write **Boolean**.

*expression*.**AllowDeletingRows**

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

## Example

This example locks all cells on Sheet1, and then it enables the insertion and deletion of rows, and then protects Sheet1.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    'Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to delete rows while Sheet1 is protected.  
    ptProtSheet1.AllowDeletingRows = True  
  
    ' Allows user to insert rows while Sheet1 is protected.  
    ptProtSheet1.AllowInsertingRows = True  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
End Sub
```





# AllowDeletions Property

**GroupLevel object:** Returns or sets a **Boolean** that represents whether the user can delete records from the specified group. Set this property to **False** to prevent users from deleting records from a group. The default value is **True**. Read/write.

**PivotView object:** Returns or sets a **Boolean** that represents whether the user can delete a record from detail area of a PivotTable list. Set this property to **True** to allow the user to delete records. The default value is **False**. Read/write.

*expression*.**AllowDeletions**

*expression* Required. An expression that returns a **GroupLevel** or **PivotView** object.

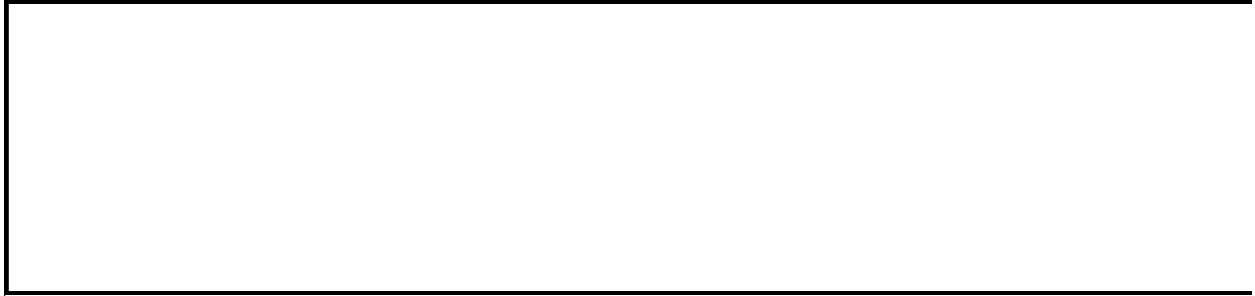
## Remarks

Any records that are deleted from the detail area of a PivotTable list are also deleted from the source database. The user cannot delete records from the detail area of the PivotTable list if the source recordset does not allow deletions.

## Example

This example allows the user to delete records from the detail area of PivotTable1.

```
PivotTable1.ActiveView.AllowDeletions = True
```



# AllowDetails Property

-  
Specifies whether the user can expand an inner member of the specified PivotTable list to display detail records. The default value is **True**. Read/write **Boolean**.

*expression*.**AllowDetails**

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

## Remarks

When this property is **False**, inner members do not display expand indicators, the **Expand** command is disabled for most aggregates, and double-clicking an aggregate does not display detail data. If this property is **False**, the user can still view details programmatically.

## Example

This example disables detail viewing for the PivotTable list.

```
PivotTable1.AllowDetails = False
```



# AllowEdits Property

**GroupLevel object:** Returns or sets a **Boolean** that represents whether the user can edit records in the specified group. Set this property to **False** to prevent users from editing records in a group. The default value is **True**. Read/write.

**PivotView object:** Returns or sets a **Boolean** that represents whether the user can edit cells in the detail area of a PivotTable list. Set this property to **True** to allow the user to edit cells in the detail area. The default value is **False**. Read/write.

*expression*.**AllowEdits**

*expression* Required. An expression that returns a **GroupLevel** or **PivotView** object.

## Remarks

Changing a cell value results in a corresponding change in the source database. Fields that are marked as read-only in the source database cannot be edited in the PivotTable list.



## Example

This example enables editing in the detail area of PivotTable1.

```
PivotTable1.ActiveView.AllowEdits = True
```



▼ [Show All](#)

# AllowFiltering Property

## ► [AllowFiltering property as it applies to the \*\*Protection\*\* object.](#)

Determines whether AutoFilter can be enabled or disabled when the worksheet has been protected. This property has no effect if the **Protection** object's **Enabled** property is set to **False**. Setting this property to **True** disables the command on the toolbar. Users can still filter data if AutoFilter is enabled before this property is set to **False**. Read/write **Boolean**.

*expression*.**AllowFiltering**

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

## ► [AllowFiltering property as it applies to the \*\*PivotTable\*\* object.](#)

Determines whether a field can be added to or removed from the filter area of a PivotTable list, and whether the AutoFilter command on the toolbar is enabled. The default value is **True**. Read/write **Boolean**.

*expression*.**AllowFiltering**

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

## Example

### ► [As it applies to the \*\*Protection\*\* object.](#)

This example locks all cells on Sheet1, enables the insertion and deletion of columns, disables the AutoFilter command on the toolbar, and then protects Sheet1.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    'Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to delete columns while Sheet1 is protected.  
    ptProtSheet1.AllowDeletingColumns = True  
  
    ' Allows user to insert columns while Sheet1 is protected.  
    ptProtSheet1.AllowInsertingColumns = True  
  
    ' Disable the AutoFilter command on the toolbar.  
    ptProtSheet1.AllowFiltering = False  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
End Sub
```

### ► [As it applies to the \*\*PivotTable\*\* object.](#)

This example disables filtering for the PivotTable list.

```
PivotTable1.AllowFiltering = False
```



# AllowFormattingColumns Property

-

Returns or sets whether or not columns can be resized on a protected worksheet. Set this property to **True** to enable columns to be resized when the worksheet is protected. The default value is **False**. Read/write **Boolean**.

*expression*.**AllowFormattingColumns**

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

## Example

This example locks all cells on Sheet1, and then it enables the user to resize rows and columns, and then protects Sheet1.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    ' Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to resize rows while Sheet1 is protected.  
    ptProtSheet1.AllowFormattingRows = True  
  
    ' Allows user to resize columns while Sheet1 is protected.  
    ptProtSheet1.AllowFormattingColumns = True  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
End Sub
```



# AllowFormattingRows Property

-

Returns or sets whether or not rows can be resized on a protected worksheet. Set this property to **True** to enable rows to be resized when the worksheet is protected. The default value is **False**. Read/write **Boolean**.

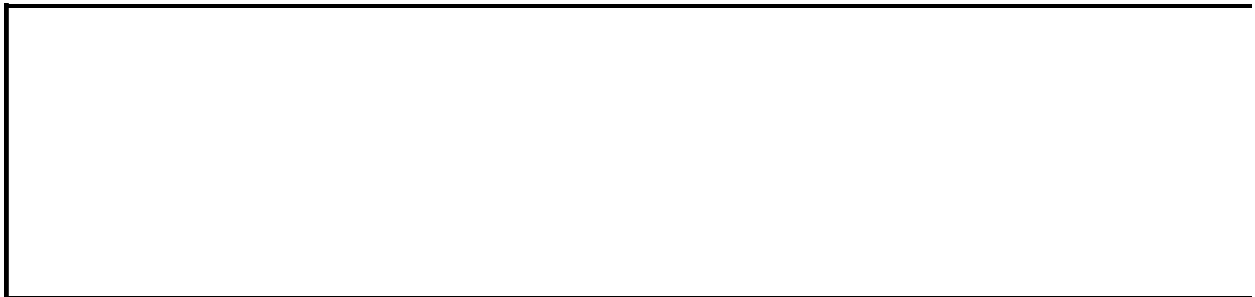
*expression*.**AllowFormattingRows**

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

## Example

This example locks all cells on Sheet1, and then it enables the user to resize rows and columns, and then protects Sheet1.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    ' Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to resize rows while Sheet1 is protected.  
    ptProtSheet1.AllowFormattingRows = True  
  
    ' Allows user to resize columns while Sheet1 is protected.  
    ptProtSheet1.AllowFormattingColumns = True  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
  
End Sub
```





# AllowGrouping Property

**True** if the user is able to group fields on the row axis or the column axis in the specified PivotTable list. If this property is set to **False**, the grouping buttons are disabled but the user can still group fields programmatically. The default value is **True**. Read/write **Boolean**.

*expression*.**AllowGrouping**

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

## Remarks

When this property is set to **False**, pivoting is disabled. This means that membership in the row and column areas is fixed; you cannot drag a field to or from these areas. All commands related to grouping are disabled, including the following commands:

- **Move to Column Area**
- **Move to Detail**, if the selection is a field in the row or column area
- **Move to Filtering**
- **Move to Row Area**

If **False**, the report layout is frozen, but the user can still filter the data.

## Example

This example disables field grouping for the PivotTable list.

```
PivotTable1.AllowGrouping = False
```



# AllowHeadingRename Property

-  
Specifies whether row and column headers in a protected worksheet can be customized. Set this property to **True** to customize the column headers of a protected worksheet. The default value is **False**. Read/write **Boolean**.

*expression*.**AllowHeadingRename**

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

## Remarks

Setting the **Caption** property of a row or column heading after setting this property to **False** results in a run-time error. You must set the **Enabled** property of the **Protection** object to **True** for this property to take effect.

## Example

This example locks all cells on Sheet1, enables the insertion and deletion of columns, disables the customization of row and column headings, and then protects Sheet1.

```
Sub Protect_Worksheet()  
  
    Dim ptProtSheet1  
  
    'Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to delete columns while Sheet1 is protected.  
    ptProtSheet1.AllowDeletingColumns = True  
  
    ' Allows user to insert columns while Sheet1 is protected.  
    ptProtSheet1.AllowInsertingColumns = True  
  
    ' prevent row and column headings from being customized.  
    ptProtectSheet1.AllowHeadingRename = False  
  
    ' Protect Sheet1.  
    ptProtectSheet1.Enabled = True  
  
End Sub
```



# AllowInsertingColumns Property

-  
Specifies whether a worksheet column can be inserted when the worksheet has been protected. The default value is **False**, but this property has no effect if the **Enabled** property of the **Protection** object is set to **False**. Read/write **Boolean**.

*expression*.**AllowInsertingColumns**

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

## Example

This example locks all cells on Sheet1, and then it enables the insertion and deletion of columns, and then protects Sheet1.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    'Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to delete columns while Sheet1 is protected.  
    ptProtSheet1.AllowDeletingColumns = True  
  
    ' Allows user to insert columns while Sheet1 is protected.  
    ptProtSheet1.AllowInsertingColumns = True  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
End Sub
```





# AllowInsertingRows Property

-  
Specifies whether a worksheet row can be inserted when the worksheet has been protected. The default value is **False**, but this property has no effect if the [Enabled](#) property of the **Protection** object is set to **False**. Read/write **Boolean**.

*expression*.**AllowInsertingRows**

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

## Example

This example locks all cells on Sheet1, and then it enables the insertion and deletion of rows, and then protects Sheet1.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    'Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to delete rows while Sheet1 is protected.  
    ptProtSheet1.AllowDeletingRows = True  
  
    ' Allows user to insert rows while Sheet1 is protected.  
    ptProtSheet1.AllowInsertingRows = True  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
End Sub
```



# AllowLayoutEvents Property

-

Set this property to **True** to enable the **AfterLayout** event. The default value is **False**. Read/write **Boolean**.

*expression*.**AllowLayoutEvents**

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

## Example

This example enables all events for ChartSpace1.

```
Sub Window_Onload()  
    ' Allow the AfterLayout event to fire.  
    ChartSpace1.AllowLayoutEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to fire as each data point is rendered.  
    ChartSpace1.AllowPointRenderEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to fire as each chart element is rendered.  
    ' The AfterFinalRender event will fire after  
    ' all chart elements have been rendered.  
    ChartSpace1.AllowRenderEvents = True  
  
    ' Allow the BeforeScreenTip event to fire.  
    ChartSpace1.AllowScreenTipEvents = True  
  
End Sub
```



# AllowMultiFilter Property

-

Returns or sets whether or not the user can select multiple items when the specified field set is in the filter area of a PivotTable list. The default value is **True**. Read/write **Boolean**.

*expression*.**AllowMultiFilter**

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

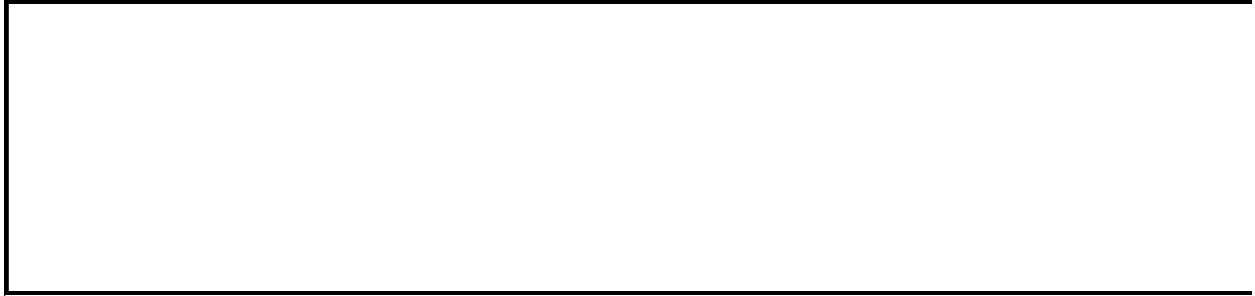
## Remarks

Set this property to **False** to emulate the behavior of a field set in the filter area of the Microsoft Office 2000 PivotTable Component.

## Example

The following example disables the selection of multiple items in the Merchant field when it is in the filter area of PivotTable1.

```
PivotTable1.ActiveView.FieldSets("Merchant").AllowMultiFilter = Fals
```



# AllowPointRenderEvents Property

-

Set this property to **True** to enable the BeforeRender and AfterRender events to be called as each data point is rendered. The default value is **False**. Read/write **Boolean**.

*expression*.**AllowPointRenderEvents**

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



## Example

This example enables all events for ChartSpace1.

```
Sub Window_Onload()  
    ' Allow the AfterLayout event to be called.  
    ChartSpace1.AllowLayoutEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to be called as each data point is rendered.  
    ChartSpace1.AllowPointRenderEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to be called as each chart element is rendered.  
    ' The AfterFinalRender event will fire after  
    ' all chart elements have been rendered.  
    ChartSpace1.AllowRenderEvents = True  
  
    ' Allow the BeforeScreenTip event to be called.  
    ChartSpace1.AllowScreenTipEvents = True  
  
End Sub
```



# AllowPropertyToolbox Property

-

Determines whether the user can display the Commands and Options dialog box at run time. Setting the **AllowPropertyToolbox** property to **False** disables the Commands and Options button on the toolbar. The default value is **True**.  
Read/write **Boolean**.

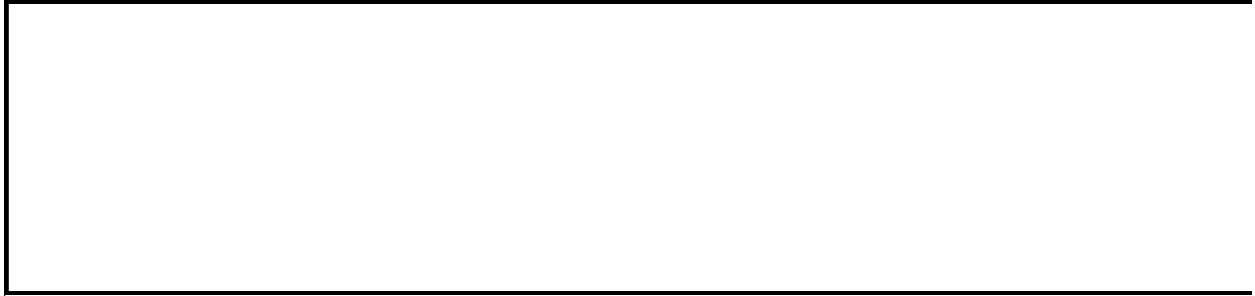
*expression*.**AllowPropertyToolbox**

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

## Example

This example disables the Commands and Options button on the Spreadsheet1 toolbar at run time.

```
Spreadsheet1.AllowPropertyToolbox = False
```



# AllowRenderEvents Property

-

Set this property to **True** to enable the BeforeRender, AfterRender, and AfterFinalRender events. The default value is **False**. Read/write **Boolean**.

*expression*.**AllowRenderEvents**

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

## Example

This example enables all events for ChartSpace1.

```
Sub Window_Onload()  
    ' Allow the AfterLayout event to fire.  
    ChartSpace1.AllowLayoutEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to fire as each data point is rendered.  
    ChartSpace1.AllowPointRenderEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to fire as each chart element is rendered.  
    ' The AfterFinalRender event will fire after  
    ' all chart elements have been rendered.  
    ChartSpace1.AllowRenderEvents = True  
  
    ' Allow the BeforeScreenTip event to fire.  
    ChartSpace1.AllowScreenTipEvents = True  
End Sub
```



# AllowScreenTipEvents Property

-

Set this property to **True** to enable the BeforeScreenTip event. The default value is **False**. Read/write **Boolean**.

*expression*.**AllowScreenTipEvents**

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

## Remarks

The BeforeScreenTip event allows you to modify ScreenTips before they are displayed.

## Example

This example enables all events for ChartSpace1.

```
Sub Window_Onload()  
    ' Allow the AfterLayout event to be called.  
    ChartSpace1.AllowLayoutEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to fire as each data point is rendered.  
    ChartSpace1.AllowPointRenderEvents = True  
  
    ' Allow BeforeRender and AfterRender events  
    ' to be called as each chart element is rendered.  
    ' The AfterFinalRender event will be called after  
    ' all chart elements have been rendered.  
    ChartSpace1.AllowRenderEvents = True  
  
    ' Allow the BeforeScreenTip event to be called.  
    ChartSpace1.AllowScreenTipEvents = True  
  
End Sub
```





# AllowSorting Property

-  
Specifies whether a worksheet can be sorted when the worksheet has been protected. The default value is **False**, but this property has no effect if the [Enabled](#) property of the **Protection** object is set to **False**. Read/write **Boolean**.

*expression*.**AllowSorting**

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

## Example

This example locks all cells on Sheet1, enables the filtering and sorting of rows and columns, and then protects the worksheet.

```
Sub Protect_Worksheet()  
    Dim ptProtSheet1  
  
    'Lock all cells on the worksheet.  
    Spreadsheet1.Worksheets("Sheet1").Cells.Locked = True  
  
    Set ptProtSheet1 = Spreadsheet1.Worksheets("Sheet1").Protection  
  
    ' Allows user to filter rows while Sheet1 is protected.  
    ptProtSheet1.AllowFiltering = True  
  
    ' Allows user to sort rows and columns while Sheet1 is protected  
    ptProtSheet1.AllowSorting = True  
  
    ' Protect Sheet1.  
    ptProtSheet1.Enabled = True  
End Sub
```



# AllPageFields Property

-  
Returns the [AllPageFields](#) collection for the data source control. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).



# AlternateDataSource Property

-

Returns or sets the ID of the alternate data source (another complex data-bound control that will be used as the data source). Read/write **String**.

---

*expression*.**AlternateDataSource**

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

# AlternateRowColor Property

-

Returns or sets a **String** that represents the color to use for every other row in the specified group. Read/write.

*expression*.**AlternateRowColor**

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

--

# AlwaysIncludeInCube Property

-

Returns or sets whether the specified field set is always included in the PivotTable list's local cache. Set this property to **True** to ensure that the specified field set is included in the local cache. The default value is **False**. Read/write **Boolean**.

*expression*.**AlwaysIncludeInCube**

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

## Remarks

Setting this property to **True** for a field set ensures that calculated totals that depend on the field set will calculate correctly.

# AmbientLightIntensity Property

-

Returns or sets a **Double** specifying the percentage of ambient light illuminating a three-dimensional chart. Valid settings range from 0 to 1. Read/write.

*expression*.**AmbientLightIntensity**

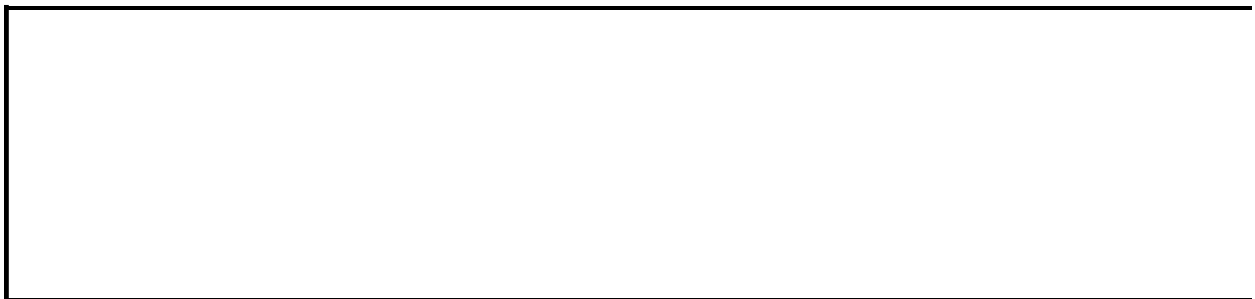
*expression* Required. An expression that returns a **ChChart** object.



## Example

This example converts the first chart in Chartspace1 to a 3-D Bar chart and sets the lighting options for the chart.

```
Sub Format3DLightSources()  
  
    Dim cht3DBar  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DBar = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Bar chart.  
    cht3DBar.Type = chChartTypeBar3D  
  
    ' Set the intensity of the ambient light.  
    cht3DBar.AmbientLightIntensity = 0.7  
  
    ' Set the inclination of the directional light source.  
    cht3DBar.DirectionallightInclination = 35  
  
    ' Set the intensity of the directional light source.  
    cht3DBar.DirectionallightIntensity = 0.8  
  
    ' Set the rotation of the directional light source.  
    cht3DBar.DirectionallightRotation = 120  
  
End Sub
```



# Amount Property

-

Returns or sets the error amount for fixed-value and percentage error bars. You specify data-bound error bar amounts by using the [SetData](#) method. Read/write **Double**.

*expression*.**Amount**

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

## Remarks

This value is mathematically correct, meaning that 5% is represented as 0.05 and not 5. An error bar with the fixed amount of 5 will become 500% when changed to a percentage.

## Example

This example adds error bars to all of the series in the first chart of ChartSpace1 and then sets the error amount.

```
Sub Add_Error_Bars()  
    Dim ebErrorBars  
    Dim serChartSeries  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Loop through all of the series in the first chart  
    ' of ChartSpace1.  
    For Each serChartSeries in ChartSpace1.Charts(0).SeriesCollection  
        ' Add error bars to the current series.  
        Set ebErrorBars = serChartSeries.ErrorBarsCollection.Add  
  
        ' Set the error bars to be a percentage of the value.  
        ebErrorBars.Type = chConstants.chErrorBarTypePercent  
  
        ' Set the percentage amount.  
        ebErrorBars.Amount = 0.05  
    Next  
End Sub
```



▼ [Show All](#)

# Application Property

-

► [Application property as it applies to all objects in the Applies To list except for the \*\*OWCLanguageSettings\*\* object.](#)

Returns a **Spreadsheet** object that represents the spreadsheet control.

*expression*.**Application**

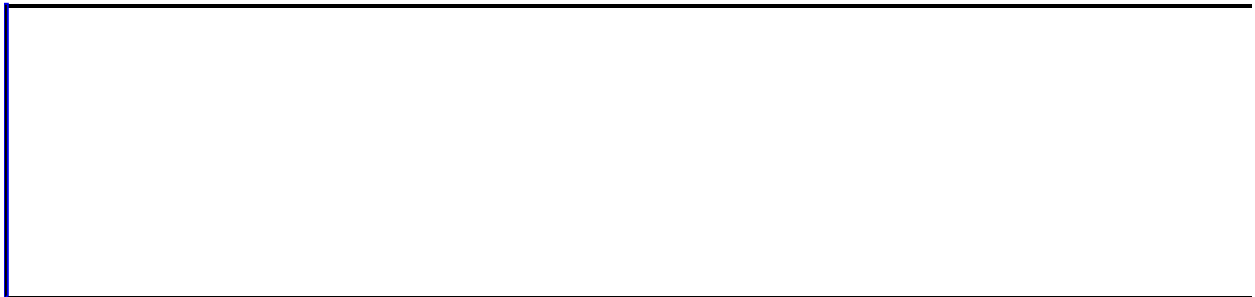
*expression* Required. An expression that returns one of the above objects.

► [Application property as it applies to the \*\*OWCLanguageSettings\*\* object.](#)

Returns an **Object** that represents the specified Microsoft Office Web Component.

*expression*.**Application**

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



# AspectRatio Property

-

Returns or sets a **Long** specifying the ratio of height to the width of the specified three-dimensional chart. Setting this property to a value greater than 100 will make a chart which is taller than it is wide, while a value less than 100 will make a chart wider than it is tall. Valid settings range from 0 to 500. Read/write.

*expression*.**AspectRatio**

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

## Remarks

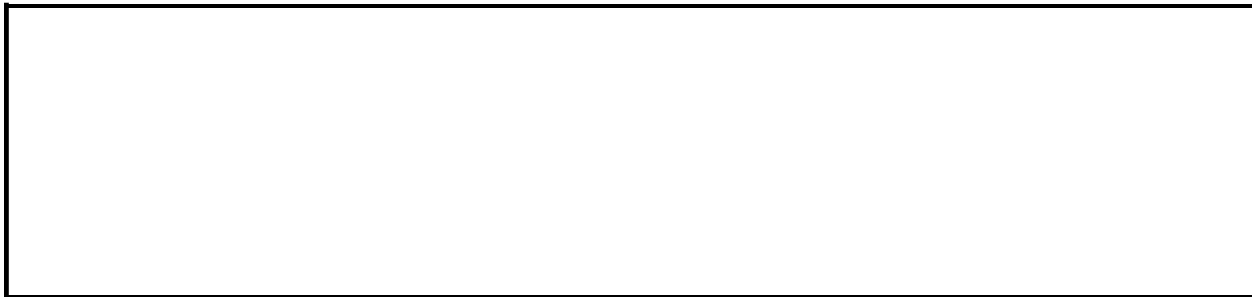
This property has no effect on a 3-D Pie chart. Use the **Thickness** property to increase the thickness of a 3-D Pie chart.



## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then increases the width and depth of the chart.

```
Sub Format3DDepthWidth()  
    Dim cht3DColumn  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Column chart.  
    cht3DColumn.Type = chChartTypeColumnClustered3D  
  
    ' Increase the depth of the chart in relation  
    ' to it's width.  
    cht3DColumn.ChartDepth = 125  
  
    ' Increase the width of the chart in relation  
    ' to it's height.  
    cht3DColumn.AspectRatio = 80  
  
End Sub
```



# AutoFilter Property

-

Returns the [AutoFilter](#) object for the specified worksheet.

*expression*.**AutoFilter**

*expression* An expression that returns a **Worksheet** object.

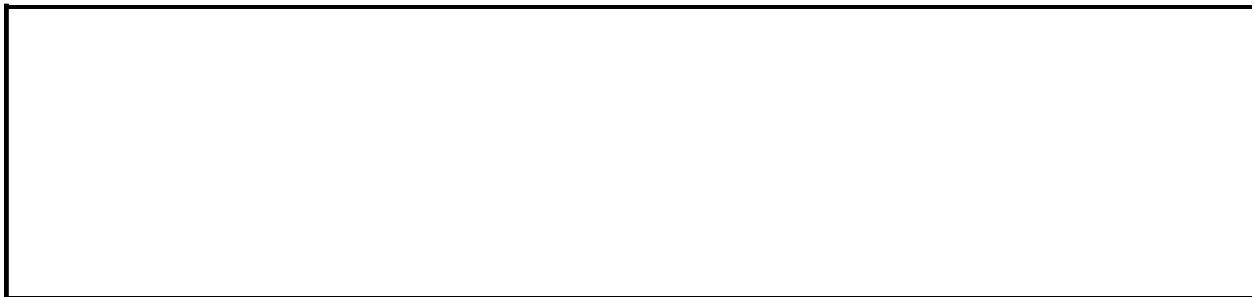
## Remarks

Do not confuse this property with the [AutoFilter](#) method. This property returns the **AutoFilter** object for a given worksheet, whereas the **AutoFilter** method applies to a **Range** object and turns on the AutoFilter.

## Example

This example turns on the AutoFilter for the range A1:C20, sets filters for columns A and C, and then applies the filters.

```
Sub Apply_AutoFilter()  
    Dim afFilters  
    Dim afCol1  
    Dim afCol3  
  
    ' Turn on AutoFilter.  
    Spreadsheet1.Worksheets("Sheet1").Range("A1:C20").AutoFilter  
  
    ' Set a variable to the AutoFilter object.  
    Set afFilters = Spreadsheet1.Worksheets("sheet1").AutoFilter  
  
    Set afCol1 = afFilters.Filters(1)  
    Set afCol3 = afFilters.Filters(3)  
  
    ' Add a criteria that excludes blue from column A.  
    afCol1.Criteria.Add "blue"  
  
    ' Add a criteria that excludes green from column A.  
    afCol1.Criteria.Add "green"  
  
    ' Add a criteria that excludes yellow from column C.  
    afCol3.Criteria.Add "yellow"  
  
    ' Apply the criteria.  
    afFilters.Apply  
End Sub
```



# AutoFilterMode Property

-

Returns **True** if the AutoFilter drop-down arrows are currently displayed. You can set this property to **False** to hide the arrows, but you cannot set it to **True**. Use the [AutoFilter](#) method to filter a list and display the drop-down arrows. Read/write **Boolean**.

*expression*.**AutoFilterMode**

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

## Remarks

If the AutoFilter drop-down arrows are visible but no rows are currently filtered (all rows are visible), the **AutoFilterMode** property is **True** and the [FilterMode](#) property is **False**.

## Example

This example turns off the AutoFilter for the active worksheet.

```
Spreadsheet1.ActiveSheet.AutoFilterMode = False
```



# AutoFit Property

-

**Spreadsheet** or **PivotTable** objects: **True** if the overall size of the spreadsheet or PivotTable list is determined by the number of visible columns and rows. The row height and column width cannot exceed the value of the spreadsheet's or PivotTable list's [MaxHeight](#) and [MaxWidth](#) properties. The default value is **True**. If the **AutoFit** property is set to **False**, the overall size of the spreadsheet or PivotTable list is set based on its [Width](#) and [Height](#) properties. Read/write **Boolean**.

**PivotAxisMember** or **PivotTotal** objects: **True** if the width of the total is set automatically. The default value is **True**. When this property is set to **True**, any layout change also updates the **Width** and **Height** property values. When you change the **Width** and **Height** values programmatically, the value of the **AutoFit** property is set to **False**. Read/write **Boolean**.

*expression*.**AutoFit**

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



## Example

This example disables automatic sizing for the PivotTable list and then sets its width.

```
Sub DisableAutoFit()  
    PivotTable1.AutoFit = False  
    PivotTable1.Object.Width = 8000  
End Sub
```



# Axes Property

-

Returns the **ChAxes** collection for the specified chart. Use the **Axes** property to set the properties for a chart axis. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

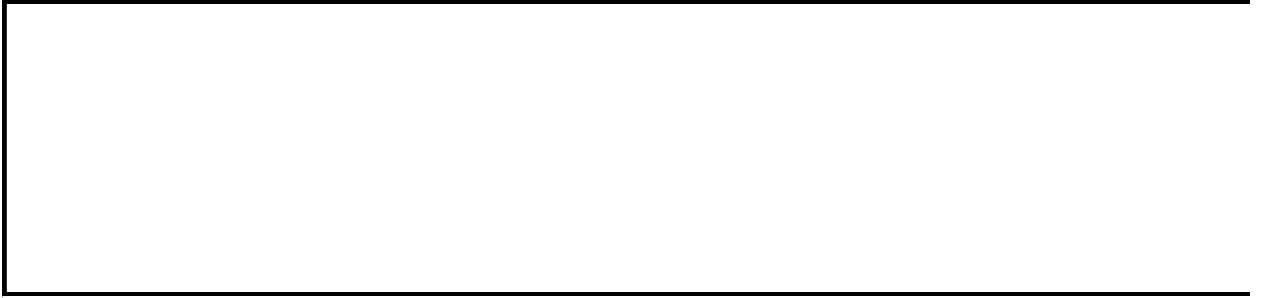
*expression*.**Axes**

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

## Example

This example adds a title to, and changes the font of the value and category axes of the first chart in ChartSpace1.

```
Sub Format_Chart_Axes()  
  
    Dim axCategoryAxis  
    Dim axValueAxis  
  
    ' Set a variable to the Category (X) axis.  
    Set axCategoryAxis = ChartSpace1.Charts(0).Axes(0)  
  
    ' Set a variable to the Value (Y) axis.  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(1)  
  
    ' The following two lines of code enable, and then  
    ' set the title for the category axis.  
    axCategoryAxis.HasTitle = True  
    axCategoryAxis.Title.Caption = "Sales by Quarter"  
  
    ' The following three lines of code set the font  
    ' for the values displayed on the category axis.  
    axCategoryAxis.Font = "Tahoma"  
    axCategoryAxis.Font.Size = 8  
    axCategoryAxis.Font.Bold = True  
  
    ' The following two lines of code enable, and then  
    ' set the title for the value axis.  
    axValueAxis.HasTitle = True  
    axValueAxis.Title.Caption = "Dollars ($)"  
  
    ' The following three lines of code set the font  
    ' for the values displayed on the value axis.  
    axValueAxis.Font = "Tahoma"  
    axValueAxis.Font.Size = 8  
    axValueAxis.Font.Bold = True  
end sub
```



▼ [Show All](#)

# Axis Property

▸ [Axis property as it applies to the \*\*PivotResultField\*\* and \*\*PivotResultGroupField\*\* objects.](#)

Returns a **PivotResultAxis** object that represents the parent axis for the specified **PivotResultField** or **PivotResultGroupField**.

*expression*.Axis

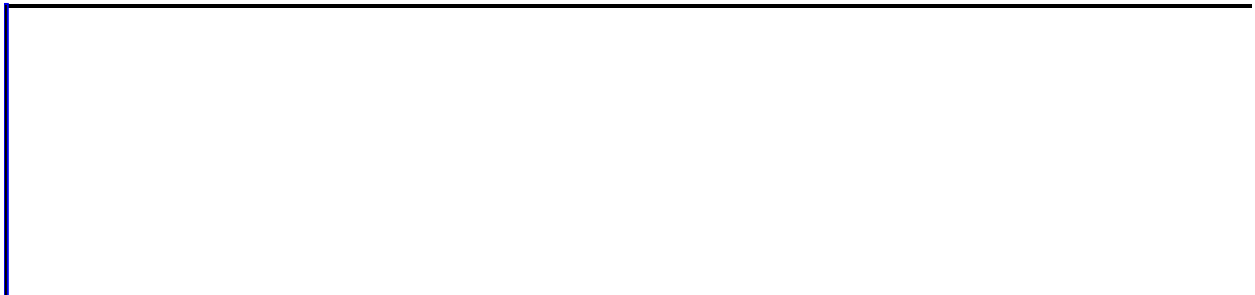
*expression* Required. An expression that returns one of the above objects.

▸ [Axis property as it applies to the \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotPageMember\*\*, and \*\*PivotRowMember\*\* objects.](#)

Returns a **PivotResultGroupAxis** object that represents the parent axis for the specified object.

*expression*.Axis

*expression* Required. An expression that returns one of the above objects.



# BackColor Property

Returns or sets the background color for the specified object or area. When you set this property, you can use either a **Long** value representing a red-green-blue (RGB) color value or a **String** value representing a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (for example, red is RGB(255, 0, 0)). Read/write **Variant**.

*expression*.**BackColor**

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

## Remarks

This property always returns the color as a **Long** value representing an RGB color value.



## Example

This example sets the font size, foreground color, and background color for the title bar in PivotTable1 .

```
Sub Format_Titlebar()  
    Dim vwView  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' Set the background color of the title bar.  
    vwView.Label.BackColor = "DarkSalmon"  
  
    ' Set the font size of the title bar.  
    vwView.Label.Font.Size = 16  
  
    ' Set the foreground color of the title bar.  
    vwView.Label.ForeColor = "Sienna"  
End Sub
```



# BackWall Property

-

Returns a **ChSurface** object that represents the back wall of a three-dimensional chart. Use the properties and methods of the returned **ChSurface** object to format the back wall of the specified chart.

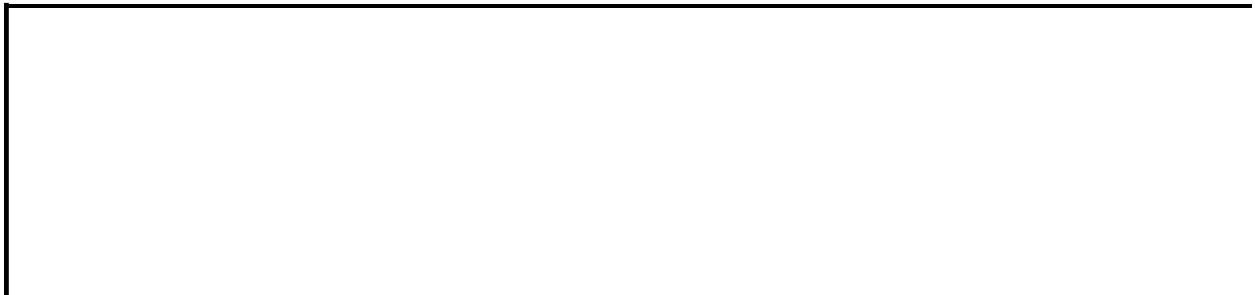
*expression*.**BackWall**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then formats the back wall, side wall, and floor of the chart.

```
Sub FormatWallsFloor()  
  
    Dim cht3DColumn  
    Dim chConstants  
    Dim paPlotArea  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Set a variable to the plot area.  
    Set paPlotArea = cht3DColumn.PlotArea  
  
    ' Change the chart to a 3D Column chart.  
    cht3DColumn.Type = chConstants.chChartTypeColumnClustered3D  
  
    ' Format the back wall of the chart.  
    paPlotArea.BackWall.Interior.SetSolid "Yellow"  
    paPlotArea.BackWall.Thickness = 5  
  
    ' Format the side wall of the chart.  
    paPlotArea.SideWall.Interior.SetSolid "Yellow"  
    paPlotArea.SideWall.Thickness = 5  
  
    ' Format the floor of the chart.  
    paPlotArea.Floor.Interior.SetSolid "Blue"  
    paPlotArea.Floor.Thickness = 5  
  
End Sub
```



# BaseName Property

-  
Returns the name of the specified field as it appears in the source database.  
Read-only **String**.

*expression*.**BaseName**

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

## Remarks

A field can have several names, as shown in the following table.

Name	Description
Caption	The name the user sees.
Name	The name used to identify the field in code.
DataField	The name of the field in the underlying recordset. (This is not necessarily the same as the base name; you can use a different name if there is a naming conflict or if you want to make the name easier to use and remember.)
BaseName	The name of the field in the original source database.

# Begin Property

-

Returns a **ChSegmentBoundary** object that represents the beginning of a segment boundary on a format map.

*expression*.**Begin**

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

## Remarks

Use the **Value** property of the returned **ChSegmentBoundary** object to set the beginning value for the specified segment of the format map. Use the **Interior**, **Line**, and **Border** properties to format the segment boundary.

## Example

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created that highlights the bottom 10% of the values in red and the top 20% of values in green.

```
Sub Window_Onload()  
  
    Dim serseries1  
    Dim segBottom10Pct  
    Dim segTop20Pct  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=ServerName  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind Chartspace1 to the Quantity  
    ' in the Order details table.  
    ChartSpace1.SetData chConstants.chDimCategories, chConstants.chDimValues  
    ChartSpace1.SetData chConstants.chDimValues, chConstants.chDimValues  
  
    ' Create a format map.  
    ChartSpace1.SetData chConstants.chDimFormatValues, chConstants.chDimValues  
  
    ' Set a variable to the first series in the first chart in ChartSpace1  
    Set serseries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a segment to the format map. This segment will  
    ' represent the bottom 10% of values in the chart.  
    Set segBottom10Pct = serseries1.FormatMap.Segments.Add  
  
    ' Measure the segment boundaries based upon a percentage.  
    segBottom10Pct.Begin.ValueType = chConstants.chBoundaryValuePercentage  
    segBottom10Pct.End.ValueType = chConstants.chBoundaryValuePercentage  
  
    ' Set the beginning value to 0%, and the ending value to 10%.  
    segBottom10Pct.Begin.Value = 0  
    segBottom10Pct.End.Value = 0.1  
  
    ' Format the interior of the matching values.  
    segBottom10Pct.Begin.Interior.Color = "red"
```



```
segBottom10Pct.End.Interior.Color = "red"

' Add a segment to the format map. This segment will
' represent the top 20% of values in the chart.
Set segTop20Pct = serseries1.FormatMap.Segments.Add

' Measure the segment boundaries based upon a percentage.
segTop20Pct.Begin.ValueType = chConstants.chBoundaryValuePercent
segTop20Pct.End.ValueType = chConstants.chBoundaryValuePercent

' Set the beginning value to 80%, and the ending value to 100%.
segTop20Pct.Begin.Value = 0.8
segTop20Pct.End.Value = 1

' Format the interior of the matching values.
segTop20Pct.Begin.Interior.Color = "Green"
segTop20Pct.End.Interior.Color = "Green"
```

End Sub



# Bold Property

-  
**Font** object: **True** if the specified font is bold. Read/write **Variant** (returns **Null** if some cells in the range are bold and some are not). Use the **IsNull** function to determine whether the return value is **Null**.

**PivotFont** and **ChFont** objects: Returns or sets a **Boolean** that determines whether the font for the specified object is bold. Read/write.

*expression*.**Bold**

*expression* Required. An expression that returns one of the above objects.

## Example

This example sets font properties for the specified axis in the chart workspace.

```
Sub SetAxisFont()  
  
    Dim axs  
  
    Set axs = ChartSpace1.Charts(0).Axes(1)  
  
    axs.Font.Name = "Arial"  
    axs.Font.Size = 8  
    axs.Font.Bold = True  
  
End Sub
```



# Bookmark Property

-

Returns a **Variant** that identifies the bookmark in the current ADO recordset.  
Read-only.

*expression*.**Bookmark**

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



# Border Property

-  
Returns a [ChBorder](#) object that represents the border of the specified object.

*expression*.**Border**

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

## Remarks

The border for a chart series represents the outline color on column, bar, pie, doughnut, area, and high-low-close charts. On charts with markers (such as a line chart with markers), the border represents the outline color for the markers.

## Example

This example sets the line weight of the border for the specified series in the chart workspace.

```
Sub SetBorder()  
    Dim chConstants  
    Set chConstants = ChartSpace1.Constants  
    ChartSpace1.Charts(0).SeriesCollection(0).Border _  
        .Weight = chConstants.owcLineWeightMedium  
End Sub
```



# Borders Property

-  
Returns a [Borders](#) collection that represents the four borders of a range of cells.  
Read-only.

*expression*.**Borders**

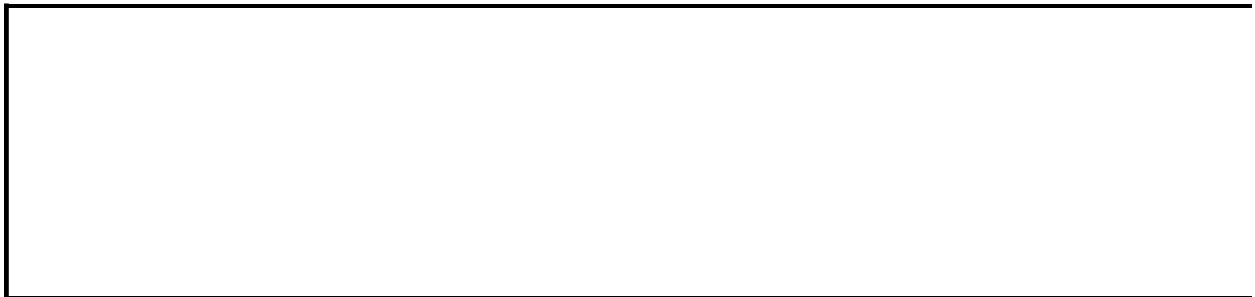
*expression* Required. An expression that returns a Range object.



## Example

This example adds a medium-weight green border to each cell in the range B5:C10 on Sheet1.

```
Sub SetBorders()  
    Dim ssConstants  
    Dim rngBorder  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Set a variable to the range to add the borders to.  
    Set rngBorder = Spreadsheet1.Worksheets("Sheet1").Range("b5:c10")  
  
    ' Set the border weight.  
    rngBorder.Borders.Weight = ssConstants.owcLineWeightMedium  
  
    ' Set the border color.  
    rngBorder.Borders.Color = "green"  
End Sub
```



▼ [Show All](#)

# Bottom Property

-

► [Bottom property as it applies to the \*\*ChPlotArea\*\* object.](#)

Returns or sets a **Long** value that represents the bottom of the specified plot area. Read/write.

*expression*.**Bottom**

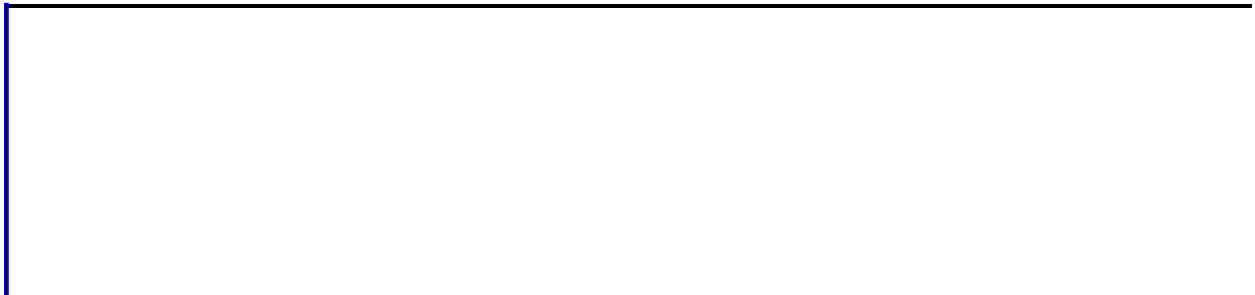
*expression* Required. An expression that returns one of the above objects.

► [Bottom property as it applies to the \*\*ChartSpace\*\*, \*\*ChAxis\*\*, \*\*ChCategoryLabel\*\*, \*\*ChChart\*\*, \*\*ChChartField\*\*, \*\*ChDataLabel\*\*, \*\*ChDataLabels\*\*, \*\*ChDropZone\*\*, \*\*ChErrorBars\*\*, \*\*ChLegend\*\*, \*\*ChLegendEntry\*\*, \*\*ChPoint\*\*, \*\*ChSeries\*\*, \*\*ChTitle\*\*, and \*\*ChTrendline\*\* objects.](#)

Returns or sets a **Long** value that represents the bottom of the specified object. Read-only.

*expression*.**Bottom**

*expression* Required. An expression that returns one of the above objects.



▼ [Show All](#)

# BottomRight Property

▸ [BottomRight property as it applies to the \*\*PivotRange\*\* object.](#)

Returns a [PivotCell](#) object that represents the cell in the lower-right corner of the specified range.

*expression*.**BottomRight**

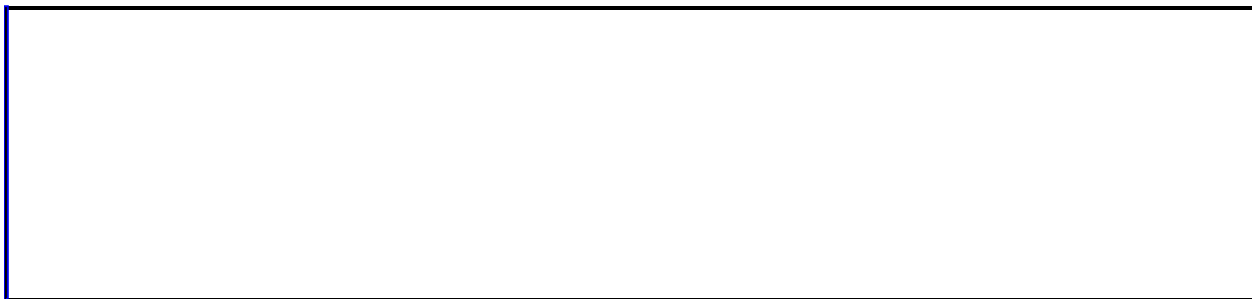
*expression* Required. An expression that returns a [PivotRange](#) object.

▸ [BottomRight property as it applies to the \*\*PivotDetailRange\*\* object.](#)

Returns a [PivotDetailCell](#) object that represents the cell in the lower-right corner of the specified detail range.

*expression*.**BottomRight**

*expression* Required. An expression that returns a [PivotDetailRange](#) object.



# BoundField Property

-

Returns the [PivotField](#) object used for data-binding information when the source is a recordset. When the source is multidimensional, this property returns **Nothing**.

*expression*.**BoundField**

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



# BubbleScale Property

-

Returns or sets a scale factor for all bubble series on the specified chart. The default value is 100, and the valid range is 0–300. The value of this property indicates the bubble size relative to the default setting. Setting the **BubbleScale** property to 300 (three times greater than the default value) produces a chart on which the bubbles appear three times the default size. Read/write **Double**.

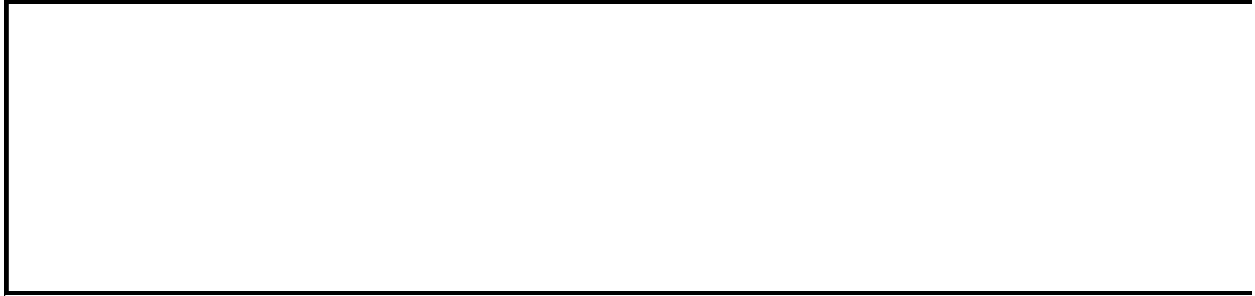
*expression*.**BubbleScale**

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

## Example

This example sets the bubble scale factor for the specified chart in the chart workspace. Note that `charts(0)` must refer to a bubble chart.

```
ChartSpace1.Charts(0).BubbleScale = 200
```





# Build Property

-

Returns a **Long** that represents the Microsoft Office Web Components build number. Read-only.

*expression*.**Build**

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

## Remarks

This property returns the same information as the **BuildNumber** property, but as a **Long** instead of as a **String**.

# BuildNumber Property

-  
Returns the Microsoft Office Web Components build number. Read-only **String**.

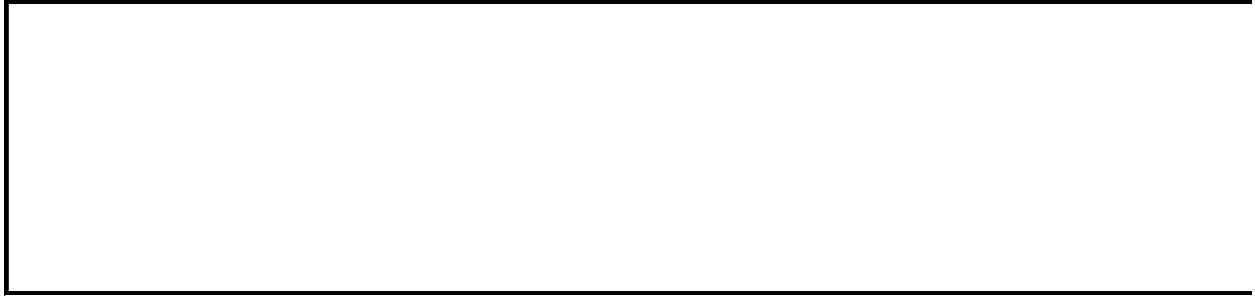
*expression*.**BuildNumber**

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

## Example

This example displays the Microsoft Office Web Components build number in the active cell of Spreadsheet1.

```
Spreadsheet1.ActiveCell.Value = "MSOWC Build " & Spreadsheet1.BuildN
```



# ButtonBorder Property

-

Returns a **ChBorder** object that represents the border of each button in the specified drop zone. Use the properties of the returned **ChBorder** object to format the border of each button in the drop zone.

*expression*.**ButtonBorder**

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

## Example

This example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```



# ButtonFont Property

-

Returns a **ChFont** object that represents the font of each button in the specified drop zone. Use the properties of the returned **ChFont** object to format the font of each button in the drop zone.

*expression*.**ButtonFont**

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

## Example

This example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```





# ButtonInterior Property

-

Returns a **ChInterior** object that represents the interior of each button in the specified drop zone. Use the properties of the returned **ChInterior** object to format the interior of each button in the drop zone.

*expression*.**ButtonInterior**

*expression* Required. An expression that returns a **ChDropZone** property.

## Remarks

By default, the chart control uses the color setting specified for 3D Objects on the Appearance tab of the Display Control Panel as the interior color for drop zone buttons.

## Example

This example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```



▼ [Show All](#)

# Calculation Property

-  
Returns or sets an [XlCalculation](#) constant specifying the calculation mode.  
Read/write.

XlCalculation can be one of these XlCalculation constants.

**xlCalculationAutomatic** Formulas are calculated automatically when precedent cells are changed.

**xlCalculationManual** Formulas are not recalculated until the user initiates a recalculation.

*expression*.**Calculation**

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

## Example

This example places Spreadsheet1 into manual calculation mode.

```
Sub ManualCalculationMode()  
    Dim ssConstants  
    Set ssConstants = Spreadsheet1.Constants  
    ' Set Spreadsheet1 to manual calculation mode.  
    Spreadsheet1.Calculation = ssConstants.xlCalculationManual  
End Sub
```



# CalculationVersion Property

-

Returns a number whose rightmost two digits are the minor version number of the calculation engine, and whose other digits (on the left) are the major version of the Microsoft Office Spreadsheet Component. Read-only **Long**.

*expression*.**CalculationVersion**

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



# CanUndo Property

**True** if there is a previous action that can be undone. Read-only **Boolean**.

*expression*.**CanUndo**

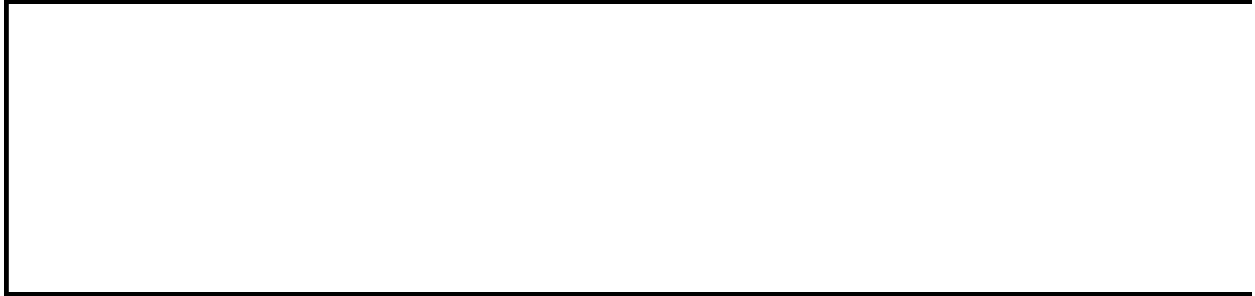
*expression* Required. An expression that returns one of the objects in the Applies To list.



## Example

This example undoes the last action performed or displays a message box if the action cannot be undone.

```
Sub Undo_Action()  
    If Spreadsheet1.CanUndo Then  
        Spreadsheet1.Undo  
    Else  
        MsgBox "Can't undo last action."  
    End If  
End Sub
```



▼ [Show All](#)

# Caption Property

▸ [Caption property as it applies to the \*\*ChSeries\*\*, \*\*ChTitle\*\*, \*\*ChTrendline\*\*, \*\*Heading\*\*, \*\*Spreadsheet\*\*, \*\*PivotField\*\*, \*\*PivotFieldSet\*\*, \*\*PivotLabel\*\*, \*\*PivotMemberProperty\*\*, \*\*PivotTotal\*\*, and \*\*TitleBar\*\* objects.](#)

Returns or sets the caption for the specified object. Read/write **String**.

*expression*.**Caption**

*expression* Required. An expression that returns one of the objects listed above.

▸ [Caption property as it applies to the \*\*ChCategoryLabel\*\*, \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotMember\*\*, \*\*PivotPageMember\*\*, \*\*PivotResultMemberProperty\*\*, and \*\*PivotRowMember\*\* objects.](#)

Returns the caption for the specified object. Read-only **String**.

*expression*.**Caption**

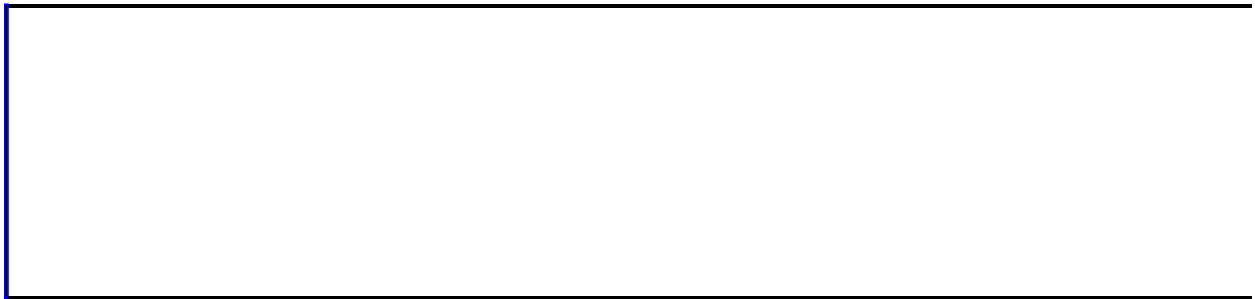
*expression* Required. An expression that returns one of the objects listed above.

## Example

► [As it applies to the \*\*Spreadsheet\*\* object.](#)

This example displays the current date and time in the title bar of Spreadsheet1.

```
Sub SetTitleBar_Caption()  
    Dim ssTitle  
  
    Set ssTitle = Spreadhseet1.TitleBar  
  
    ' Set the title bar's caption to the  
    ' current date and time.  
    ssTitle.Caption = Date & " " & Time  
  
    ' Enable the title bar.  
    ssTitle.Visible = True  
End Sub
```



# CaptionHAlignment Property

-

Returns or sets the horizontal alignment for the detail footer label in the specified PivotTable list. Read/write [PivotHAlignmentEnum](#).

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

*expression*.CaptionHAlignment

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

## Remarks

Captions can be displayed only to the left of aggregates in the detail footer. Use the **CaptionHAlignment** property to align the caption close to the aggregate (**plHAlignRight**) or to align the caption close to the left edge of the footer (**plHAlignLeft**). If there is another aggregate displayed to the left, the caption is displayed to the right of the other aggregate. If there is not enough room to display the caption, it is not visible.

## Example

This example formats totals and their captions in the PivotTable1.

```
Sub Set_Total_Alignment()  
    Dim ptConstants  
    Dim vwView  
    Dim totTotal  
  
    Set ptConstants = PivotTable1.Constants  
    Set vwView = PivotTable1.ActiveView  
  
    ' Loop through all totals in the view.  
    For Each totTotal in vwView.DataAxis.Totals  
  
        ' Set the horizontal alignment of the total to center.  
        totTotal.HAlignment = ptConstants.plHAlignCenter  
  
        ' Set the horizontal alignment of the caption to center.  
        totTotal.CaptionHAlignment = ptConstants.plHAlignCenter  
    Next  
End Sub
```



# CaptionSection Property

**True** if the specified group level has a caption section showing. Read/write **Boolean**.

*expression*.**CaptionSection**

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



## Remarks

The caption section appears on a data page only once, above all the visible records at a given group level. The caption section remains visible as you scroll through the records.

--

# CategoryLabels Property

-

Returns the collection of **ChCategoryLabels** for the specified axis. Valid only for category and timescale axes.

*expression*.**CategoryLabels**

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



# Cell Property

-

Returns a [PivotCell](#) object that indicates the location of the aggregate cell or detail cell. Use this property to return more information about the selected area in a PivotTable list.

*expression*.**Cell**

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

## Example

This example utilizes the **DoubleClick** event of PivotTable1 to display more information about a cell in the detail area of the PivotTable. This example assumes that PivotTable1 is using the Customers table form the Northwind database.

```
Sub PivotTable1_DblClick()  
    Dim ptSelection  
    Dim nRow  
    Dim rs  
  
    Set ptSelection = PivotTable1.Selection  
  
    ' If the current selection is in the  
    ' detail area of the PivotTable list.  
    If TypeName(ptSelection) = "PivotDetailRange" then  
  
        nRow = ptSelection.TopLeft.Row  
  
        ' Set a variable to the recordset of the top-left  
        ' cell in the selection.  
        Set rs = ptSelection.TopLeft.Cell.Recordset  
  
        ' Move the cursor the the correct record in the recordset.  
        rs.Move nRow  
  
        MsgBox "The row that was double-clicked was..." & String(2,  
            "Customer ID = " & rs("CustomerID") & vbCrLf & _  
            "Company Name = " & rs("CompanyName") & vbCrLf & _  
            "Contact Name = " & rs("ContactName")  
    Else  
        MsgBox "Double-click on a row!", vbExclamation  
    End If  
End Sub
```



▼ [Show All](#)

# Cells Property

▸ [Cells property as it applies to the \*\*Range\*\*, \*\*Spreadsheet\*\*, and \*\*Worksheet\*\* objects.](#)

Returns a [Range](#) object that represents the cells in the object.

*expression*.Cells

*expression* Required. An expression that returns one of the above objects.

▸ [Cells property as it applies to the \*\*PivotData\*\* and \*\*PivotRange\*\* objects.](#)

Returns a [PivotCell](#) object based on a row and column member.

*expression*.Cells(*Row*, *Column*)

*expression* Required. An expression that returns one of the above objects.

**Row** Required [PivotRowMember](#) object. The specified row.

**Column** Required [PivotColumnMember](#) object. The specified column.

## Example

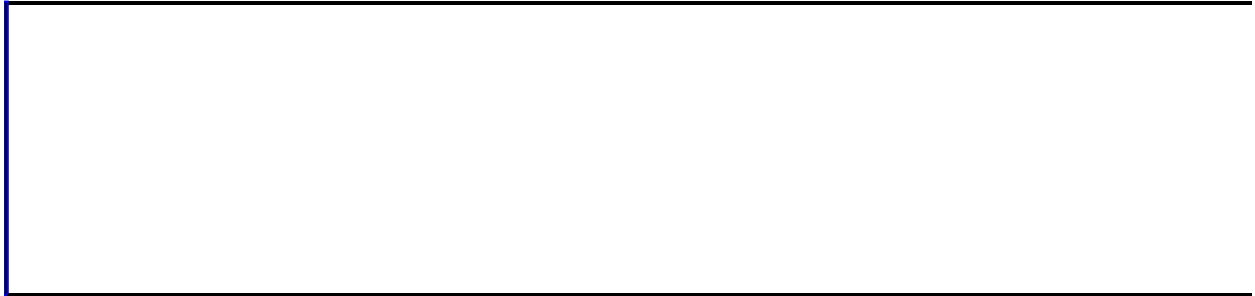
► [As it applies to the \*\*Range\*\*, \*\*Spreadsheet\*\*, and \*\*Worksheet\*\* objects.](#)

This example sets the value of cell B3 to 5.

```
Spreadsheet1.Range("A2:C4").Cells(4).Value = 5
```

This example clears all cells on Sheet1.

```
Spreadsheet1.Worksheets("Sheet1").Cells.Clear
```



# CellsEx Property

-

Returns a **PivotCell** object that represents the intersection of the specified row, column, and page members.

*expression.CellsEx(Row, Column, Page)*

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

**Row** Required **PivotRowMember** object. Specifies the row containing the returned cell.

**Column** Required **PivotColumnMember** object. Specifies the column containing the returned cell.

**Page** Required **PivotPageMember** object. Specifies the page containing the returned cell.



## Example

This example sets a variable to the total amount shipped via 2-Day shipping to postal code 28016 in North Carolina.

```
Sub Get2DayShippingToNC()  
  
    Dim ptData  
    Dim pmRowMember  
    Dim pmColMember  
    Dim pmPageMember  
    Dim pmIntersection  
    Dim dblShipTotal  
  
    Set ptData = PivotTable1.ActiveData  
  
    ' Set a variable to a row field member.  
    Set pmRowMember = ptData.RowAxis.Member.ChildAxisMembers("North  
  
    ' Set a variable to a column field member.  
    Set pmColMember = ptData.ColumnAxis.Member.ChildAxisMembers("2-D  
  
    ' Set a variable to a page field member.  
    Set pmPageMember = ptData.PageAxis.Member.ChildAxisMembers("2801  
  
    ' Set a variable to the intersection of the row, column, and pag  
    Set pmIntersection = ptData.CellsEx(pmRowMember, pmColMember, pm  
  
    ' Set a variable to the total shipping amount for the item.  
    dblShipTotal = pmIntersection.Aggregates("Shipping").Value  
  
End Sub
```



# ChartDepth Property

-

Returns or sets a **Long** specifying the depth of a three-dimensional chart in relation to it's width. Setting this property to a value a value greater than 100 will make a chart which is deeper than it is wide, while a value less than 100 will make a chart wider than it is deep. Valid settings range from 0 to 500.  
Read/write.

*expression*.**ChartDepth**

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

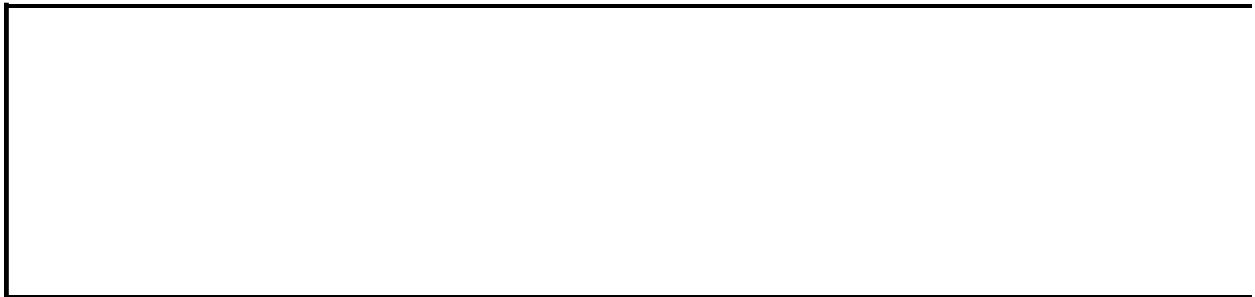
## Remarks

This property has no effect on a 3-D Pie chart.

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then increases the width and depth of the chart.

```
Sub Format_3D_Depth_Width()  
    Dim cht3DColumn  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Column chart.  
    cht3DColumn.Type = chChartTypeColumnClustered3D  
  
    ' Increase the depth of the chart in relation  
    ' to it's width.  
    cht3DColumn.ChartDepth = 125  
  
    ' Increase the width of the chart in relation  
    ' to it's height.  
    cht3DColumn.AspectRatio = 80  
  
End Sub
```



# ChartFields Property

-

Returns a **ChChartFields** object as the collection of fields that have been added to the specified drop zone.

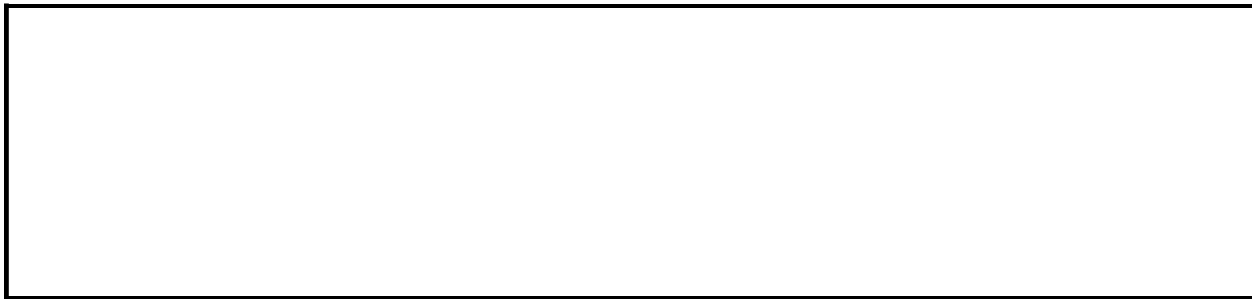
*expression*.**ChartFields**

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

## Example

This example displays the number of fields that have been added to the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' Display the number of fields that have been added to the  
    ' series drop zone.  
    MsgBox dzSeriesDropZone.ChartFields.Count  
  
End Sub
```



# ChartLayout Property

-

Returns or sets the layout for all the charts in the specified chart workspace.  
Read/write [ChartChartLayoutEnum](#).

ChartChartLayoutEnum can be one of these ChartChartLayoutEnum constants.

**chChartLayoutAutomatic**

**chChartLayoutHorizontal**

**chChartLayoutVertical**

*expression*.**ChartLayout**

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

Because the chart workspace can contain one or more charts, you can use both the **ChartLayout** and [ChartWrapCount](#) properties to specify how multiple charts are positioned. The **ChartLayout** property makes it possible to create custom chart arrangements, such as three charts positioned horizontally in a single row.

There are two distinct layout types for charts:

**ChChartLayoutHorizontal** Charts are positioned horizontally from left to right until the number of charts specified by the **ChartWrapCount** property is reached. When this occurs, a new row is created below the active row and the positioning process begins again at the left. This method continues (wrapping every **ChartWrapCount** number) until all charts have been placed.

**ChChartLayoutVertical** Charts are positioned vertically from top to bottom until the number of charts specified by the **ChartWrapCount** property is reached. When this occurs, a new column is created to the right of the active column and positioning begins again at the top. This method continues (wrapping every **ChartWrapCount** number of charts) until all charts have been placed.

## Example

This example sets the **ChartWrapCount** and **ChartLayout** properties and then adds six additional charts to the specified chart workspace.

```
Sub AddCharts()  
    Dim chtChart  
    Dim chConstants  
    Dim iCtr  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set the Chartspace so that a row or column  
    ' of charts is created for every two charts  
    ' in the chartspace.  
    ChartSpace1.ChartWrapCount = 2  
  
    ' Set the chartspace so that the carts are laid our horizontally.  
    ' Since this code adds six charts to the chartspace and the  
    ' ChrtWrapCount property has been set to wrap every two charts,  
    ' then the code results in three rows of two charts.  
    ChartSpace1.ChartLayout = chConstants.chChartLayoutHorizontal  
  
    For iCtr = 1 To 6  
        ' Add a chart to the chartspace.  
        Set chtChart = ChartSpace1.Charts.Add  
  
        ' Enable the chart title.  
        chtChart.HasTitle = True  
  
        ' Add a title to the chart that indicates the order  
        ' in which the chart was created.  
        chtChart.Title.Caption = "Chart # " & iCtr  
  
        ' Specify that the chart is to be a line chart.  
        chtChart.Type = chConstants.chChartTypeLine  
    Next  
End Sub
```



--

# Charts Property

-  
Returns the **ChCharts** collection for the specified chart workspace.

*expression*.**Charts**

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

## Remarks

By default, a new chart workspace contains no charts. After you create a new chart workspace, you must add a **Chart** object to it before you can create a chart. Use the **Add** method to create a new chart.

## Example

This example adds a chart to the specified chart workspace.

```
Set cht = ChartSpace1.Charts.Add
```



# ChartSpaceLegend Property

-

Returns a **ChLegend** object that represents the chart workspace legend. Use this property to set the properties for the chart workspace legend. Note that the **ChartSpaceLegend** property represents the legend for the entire chart workspace. Use the **Legend** property of the **ChChart** object to set the legend for individual charts within the chart workspace. Returns **Nothing** if the chart workspace does not have a legend. Read-only.

*expression*.**ChartSpaceLegend**

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

## Example

This example sets the chart workspace title and positions the chart workspace legend on the left side of the workspace.

```
Sub SetLegend()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Enable the title for the chartspace.  
    ChartSpace1.HasChartSpaceTitle = True  
  
    ' Set the title for the chartspace.  
    ChartSpace1.ChartSpaceTitle.Caption = "Monthly Sales Data"  
  
    ' Enable the legend for the chartspace.  
    ChartSpace1.HasChartSpaceLegend = True  
  
    ' Position the legend for the chartspace.  
    ChartSpace1.ChartSpaceLegend.Position = chConstants.chLegendPosi  
End Sub
```



# ChartSpaceTitle Property

-

Returns a **ChTitle** object that represents the chart workspace title. Use this property to set the properties for the chart workspace title. Note that the **ChartSpaceTitle** property represents the title for the entire chart workspace. Use the **Title** property of the **ChChart** object to set the title for individual charts within the chart workspace. Returns **Nothing** if the chart workspace does not have a title. Read-only.

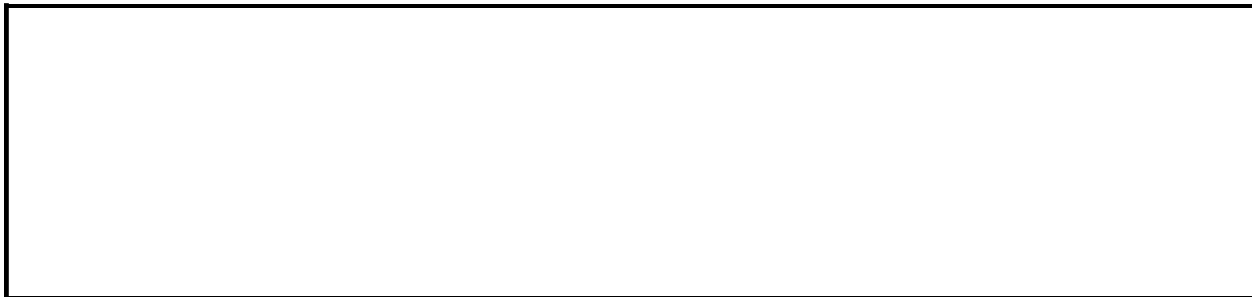
*expression*.**ChartSpaceTitle**

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

## Example

This example sets the chart workspace title and positions the chart workspace legend on the left side of the workspace.

```
Sub SetLegend()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Enable the title for the chartspace.  
    ChartSpace1.HasChartSpaceTitle = True  
  
    ' Set the title for the chartspace.  
    ChartSpace1.ChartSpaceTitle.Caption = "Monthly Sales Data"  
  
    ' Enable the legend for the chartspace.  
    ChartSpace1.HasChartSpaceLegend = True  
  
    ' Position the legend for the chartspace.  
    ChartSpace1.ChartSpaceLegend.Position = chConstants.chLegendPosi  
End Sub
```





# ChartWrapCount Property

-

Returns or sets the number of charts that are placed horizontally or vertically before wrapping occurs. For a more complete discussion of layout and wrapping, see the Help topic for the [ChartLayout](#) property. Read/write **Long**.

*expression*.**ChartWrapCount**

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

## Example

This example sets the **ChartWrapCount** and **ChartLayout** properties and then adds six additional charts to the specified chart workspace.

```
Sub AddCharts()  
    Dim chtChart  
    Dim chConstants  
    Dim iCtr  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set the Chartspace so that a row or column  
    ' of charts is created for every two charts  
    ' in the chartspace.  
    ChartSpace1.ChartWrapCount = 2  
  
    ' Set the chartspace so that the charts are laid out horizontally.  
    ' Since this code adds six charts to the chartspace and the  
    ' ChartWrapCount property has been set to wrap every two charts,  
    ' then the code results in three rows of two charts.  
    ChartSpace1.ChartLayout = chConstants.chChartLayoutHorizontal  
  
    For iCtr = 1 To 6  
        ' Add a chart to the chartspace.  
        Set chtChart = ChartSpace1.Charts.Add  
  
        ' Enable the chart title.  
        chtChart.HasTitle = True  
  
        ' Add a title to the chart that indicates the order  
        ' in which the chart was created.  
        chtChart.Title.Caption = "Chart # " & iCtr  
  
        ' Specify that the chart is to be a line chart.  
        chtChart.Type = chConstants.chChartTypeLine  
    Next  
End Sub
```



# Checked Property

-  
Returns whether the specified command is enabled. Read-only **Boolean**.

*expression*.**Checked**

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

## Remarks

The **OCCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each Web component.

# ChildAxisMembers Property

-

Returns a **PivotAxisMembers** collection that represents the children of the specified **PivotAxisMember** object.

*expression*.**ChildAxisMembers**

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

## Example

This example sets a variable to the total amount shipped via 2-Day shipping to postal code 28016 in North Carolina.

```
Sub Get2DayShippingToNC()  
  
    Dim ptData  
    Dim pmRowMember  
    Dim pmColMember  
    Dim pmPageMember  
    Dim pmIntersection  
    Dim dblShipTotal  
  
    Set ptData = PivotTable1.ActiveData  
  
    ' Set a variable to a row field member.  
    Set pmRowMember = ptData.RowAxis.Member.ChildAxisMembers("North  
  
    ' Set a variable to a column field member.  
    Set pmColMember = ptData.ColumnAxis.Member.ChildAxisMembers("2-D  
  
    ' Set a variable to a page field member.  
    Set pmPageMember = ptData.PageAxis.Member.ChildAxisMembers("2801  
  
    ' Set a variable to the intersection of the row, column, and pag  
    Set pmIntersection = ptData.CellsEx(pmRowMember, pmColMember, pr  
  
    ' Set a variable to the total shipping amount for the item.  
    dblShipTotal = pmIntersection.Aggregates("Shipping").Value  
  
End Sub
```



# ChildColumnMembers Property

-

Returns a **PivotColumnMembers** collection that represents the children of the specified **PivotColumnMember** object.

*expression*.**ChildColumnMembers**

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



## Example

This example sets a variable to the total amount shipped via 2-Day shipping to postal code 28016 in North Carolina.

```
Sub Get2DayShippingToNC()  
  
    Dim ptData  
    Dim pmRowMember  
    Dim pmColMember  
    Dim pmPageMember  
    Dim pmIntersection  
    Dim dblShipTotal  
  
    Set ptData = PivotTable1.ActiveData  
  
    ' Set a variable to a row field member.  
    Set pmRowMember = ptData.RowAxis.RowMember.ChildRowMembers("Nort  
  
    ' Set a variable to a column field member.  
    Set pmColMember = ptData.ColumnAxis.ColumnMember.ChildColumnMemb  
  
    ' Set a variable to a page field member.  
    Set pmPageMember = ptData.PageAxis.PageMember.ChildPageMembers("  
  
    ' Set a variable to the intersection of the row, column, and pag  
    Set pmIntersection = ptData.CellsEx(pmRowMember, pmColMember, pr  
  
    ' Set a variable to the total shipping amount for the item.  
    dblShipTotal = pmIntersection.Aggregates("Shipping").Value  
  
End Sub
```



# ChildLabel Property

-  
Contains the string that is displayed as the specified control's label. Read/write **String**.

*expression*.**ChildLabel**

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

---

---

# ChildMembers Property

-  
Returns a [PivotMembers](#) collection that represents the child members of the specified member.

*expression*.**ChildMembers**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# ChildPageMembers Property

-

Returns a **PivotPageMembers** collection that represents the children of the specified **PivotPageMember** object.

*expression*.**ChildPageMembers**

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

## Example

This example sets a variable to the total amount shipped via 2-Day shipping to postal code 28016 in North Carolina.

```
Sub Get2DayShippingToNC()  
  
    Dim ptData  
    Dim pmRowMember  
    Dim pmColMember  
    Dim pmPageMember  
    Dim pmIntersection  
    Dim dblShipTotal  
  
    Set ptData = PivotTable1.ActiveData  
  
    ' Set a variable to a row field member.  
    Set pmRowMember = ptData.RowAxis.RowMember.ChildRowMembers("Nort  
  
    ' Set a variable to a column field member.  
    Set pmColMember = ptData.ColumnAxis.ColumnMember.ChildColumnMemb  
  
    ' Set a variable to a page field member.  
    Set pmPageMember = ptData.PageAxis.PageMember.ChildPageMembers("  
  
    ' Set a variable to the intersection of the row, column, and pag  
    Set pmIntersection = ptData.CellsEx(pmRowMember, pmColMember, pr  
  
    ' Set a variable to the total shipping amount for the item.  
    dblShipTotal = pmIntersection.Aggregates("Shipping").Value  
  
End Sub
```



# ChildRowMembers Property

-

Returns a **PivotRowMembers** collection that represents the children of the specified **PivotRowMember** object.

*expression*.**ChildRowMembers**

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

## Example

This example sets a variable to the total amount shipped via 2-Day shipping to postal code 28016 in North Carolina.

```
Sub Get2DayShippingToNC()  
  
    Dim ptData  
    Dim pmRowMember  
    Dim pmColMember  
    Dim pmPageMember  
    Dim pmIntersection  
    Dim dblShipTotal  
  
    Set ptData = PivotTable1.ActiveData  
  
    ' Set a variable to a row field member.  
    Set pmRowMember = ptData.RowAxis.RowMember.ChildRowMembers("Nort  
  
    ' Set a variable to a column field member.  
    Set pmColMember = ptData.ColumnAxis.ColumnMember.ChildColumnMemb  
  
    ' Set a variable to a page field member.  
    Set pmPageMember = ptData.PageAxis.PageMember.ChildPageMembers("  
  
    ' Set a variable to the intersection of the row, column, and pag  
    Set pmIntersection = ptData.CellsEx(pmRowMember, pmColMember, pr  
  
    ' Set a variable to the total shipping amount for the item.  
    dblShipTotal = pmIntersection.Aggregates("Shipping").Value  
  
End Sub
```





# ChildSection Property

-

Returns a [Section](#) object that represents the child section for the specified section. This property fails if the specified section is not expanded.

*expression*.**ChildSection**

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

--

# Color Property

-

Returns or sets the primary color of the specified object. Read/write **Variant**.

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is `RGB(255, 0, 0)`).

This property always returns the color as a **Long** value representing a red-green-blue color value.

On charts, you can also use either of the following two special constants: **chColorAutomatic** (to reset the color to the default value) or **chColorNone** (to indicate no color, or transparency). However, you cannot set a font color to **chColorNone**.

## Example

This example adds a medium-weight green border to all cells in the range B5:C10 on the spreadsheet.

```
Sub SetBorder()  
    Set ssConstants = Spreadsheet1.Constants  
    Spreadsheet1.ActiveSheet.Range("b5:c10").Borders.Weight = ssCons  
    Spreadsheet1.ActiveSheet.Range("b5:c10").Borders.Color = "green"  
End Sub
```



▼ [Show All](#)

# ColorIndex Property

## ► [ColorIndex property as it applies to the \*\*Border\*\* object.](#)

Returns or sets the color of the border. The color is specified as an index value into the current color palette, or as a **XIColorIndex** constant. Read/write **Variant**.

XIColorIndex can be one of these XIColorIndex constants.

**xIColorIndexAutomatic**

**xIColorIndexNone**

*expression*.**ColorIndex**

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

## ► [ColorIndex property as it applies to the \*\*Borders\*\* object.](#)

Returns or sets the color of all four borders. The color is specified as an index value into the current color palette, or as a **XIColorIndex** constant. Returns **Null** if all four borders aren't the same color. Read/write **Variant**.

XIColorIndex can be one of these XIColorIndex constants.

**xIColorIndexAutomatic**

**xIColorIndexNone**

*expression*.**ColorIndex**

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

## ► [ColorIndex property as it applies to the \*\*Font\*\* object.](#)

Returns or sets the color of the font. The color is specified as an index value into the current color palette, or as **XIColorIndex** constant. Read/write **Variant**.

XIColorIndex can be one of these XIColorIndex constants.

**xIColorIndexAutomatic**

**xIColorIndexNone**

*expression*.**ColorIndex**

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

► [ColorIndex property as it applies to the Interior object.](#)

Returns or sets the color of the interior fill. The color is specified as an index value into the current color palette, or as a [XIColorIndex](#) constant. Read/write **Variant**.

XIColorIndex can be one of these XIColorIndex constants.

**xIColorIndexAutomatic**

**xIColorIndexNone**

























































*expression*.**ColorIndex**

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

## 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.

1		2		3		4		5		6		7	
8		9		10		11		12		13		14	
15		16		17		18		19		20		21	
22		23		24		25		26		27		28	
29		30		31		32		33		34		35	
36		37		38		39		40		41		42	
43		44		45		46		47		48		49	
50		51		52		53		54		55		56	



## Example

► [As it applies to the \*\*Font\*\* object.](#)

This example changes the font color in cell A1 on Sheet1 to red.

```
Spreadsheet1.ActiveSheet.Range("A1").Font.ColorIndex = 3
```



# Colors Property

-  
Returns or sets colors in the palette for the workbook. The palette has 56 entries, each represented by an RGB value. Read/write **Variant**.

*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.

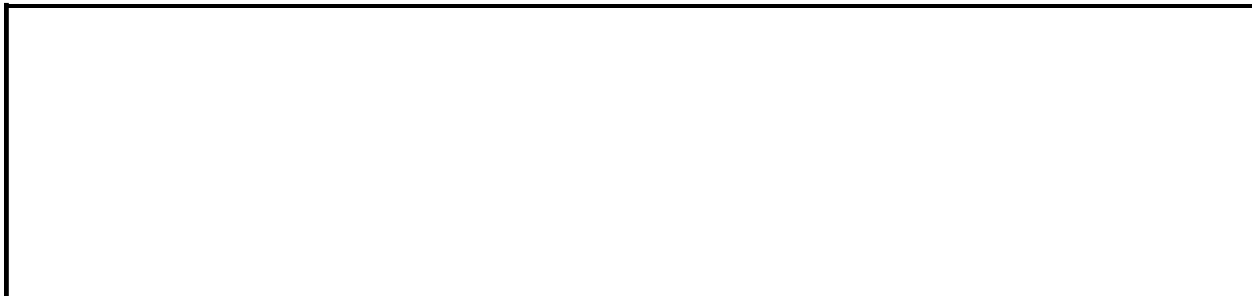
## Example

This example sets color five in the color palette for the active workbook.

```
Spreadsheet1.ActiveWorkbook.Colors(5) = RGB(255, 0, 0)
```

This example creates a table on the active worksheet in Spreadsheet1 that displays the available color palette.

```
Sub Create_Color_Table()  
    Dim avarColorArray()  
    Dim iCtr  
    Dim rngCurrent  
  
    ' Set an array variable to the colors in the color palette.  
    avarColorArray = Spreadsheet1.ActiveWorkbook.Colors  
  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Range("A1")  
  
    ' Loop through all of the colors in the array.  
    For iCtr = 1 To UBound(avarColorArray)  
        rngCurrent.Value = "Color " & iCtr  
  
        ' Set the color of a cell in column B to  
        ' the appropriate color.  
        rngCurrent.Offset(0, 1).Interior.Color _  
            = avarColorArray(iCtr)  
  
        Set rngCurrent = rngCurrent.Offset(1, 0)  
    Next  
End Sub
```



# Column Property

-  
Returns the number of the first column in the specified range. Read-only **Long**.

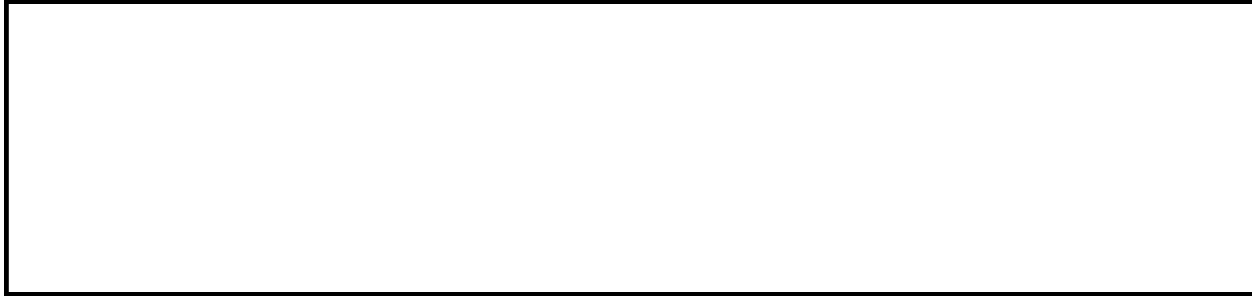
*expression*.**Column**

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

## Example

This example sets the font to bold in every other column of the visible range on the active worksheet.

```
Sub BoldColumns()  
    Dim col  
  
    For Each col In Spreadsheet1.ActiveSheet.VisibleRange.Columns  
        If col.Column Mod 2 = 0 Then col.Font.Bold = True  
    Next  
End Sub
```



# ColumnAxis Property

-  
**PivotData** object: Returns a [PivotResultColumnAxis](#) object that represents the column axis in the specified PivotTable list.

**PivotView** object: Returns a [PivotGroupAxis](#) object that represents the column axis in the specified PivotTable list.

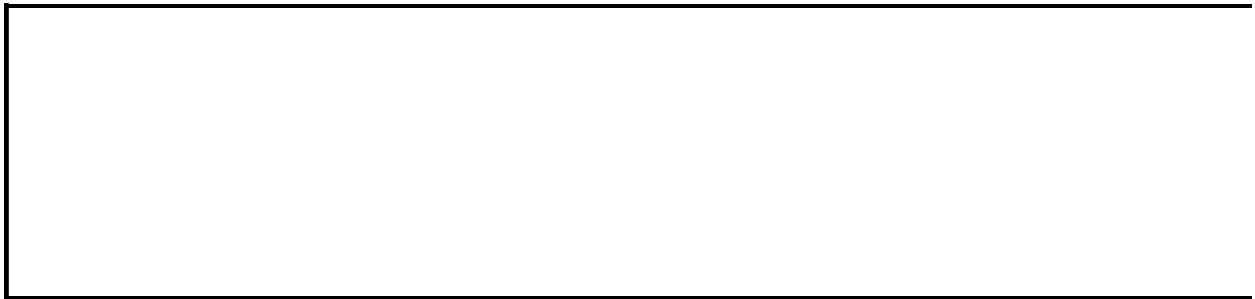
*expression*.**ColumnAxis**

*expression* Required. An expression that returns a **PivotData** or **PivotView** object.

## Example

This example inserts two field sets into the PivotTable list in the active view.

```
Sub AddFieldsToPT()  
    Dim ptView  
    Set ptView= PivotTable1.ActiveView  
    ptView.ColumnAxis.InsertFieldSet ptView.FieldSets("Store Type")  
    ptView.RowAxis.InsertFieldSet ptView.FieldSets("Promotions")  
End Sub
```



# ColumnHeadings Property

-

Returns a **Headings** collection that represents the column headings in the specified window. Use the **Caption** property to customize the column headings.

*expression*.**ColumnHeadings**

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



## Example

This example sets the creates a custom data entry sheet by disabling some user interface elements, limiting the viewable range in the active window, and customizing the row and column headings.

```
Sub Create_Datasheet()  
    Dim hdrColHeadings  
    Dim hdrRowHeadings  
    Dim wndActive  
  
    Set wndActive = Spreadsheet1.ActiveWindow  
  
    ' Hide various UI elements.  
    wndActive.DisplayWorkbookTabs = False  
    Spreadsheet1.DisplayToolbar = False  
    Spreadsheet1.DisplayScrollBars = False  
  
    ' Display the title bar and set it's caption.  
    Spreadsheet1.DisplayTitleBar = True  
    Spreadsheet1.TitleBar.Caption = "Revenue Worksheet"  
  
    ' Resize the spreadsheet component.  
    Spreadsheet1.AutoFit = True  
  
    ' Limit the viewable range of the active sheet.  
    wndActive.ViewableRange = "A1:D5"  
  
    ' Set a variable to the column headings in the active window.  
    Set hdrColHeadings = wndActive.ColumnHeadings  
  
    ' Set a variable to the row headings in the active window.  
    Set hdrRowHeadings = wndActive.RowHeadings  
  
    ' Set the headings of columns A through D.  
    hdrColHeadings(1).Caption = "Qtr 1"  
    hdrColHeadings(2).Caption = "Qtr 2"  
    hdrColHeadings(3).Caption = "Qtr 3"  
    hdrColHeadings(4).Caption = "Qtr 4"  
  
    ' Set the headings of rows 1 though 5.  
    hdrRowHeadings(1).Caption = "1996"  
    hdrRowHeadings(2).Caption = "1997"  
    hdrRowHeadings(3).Caption = "1998"  
    hdrRowHeadings(4).Caption = "1999"  
    hdrRowHeadings(5).Caption = "2000"
```

End Sub

--

# ColumnMember Property

-

Returns a [PivotColumnMember](#) object that represents the innermost member on the column axis that intersects the specified cell.

*expression*.**ColumnMember**

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



# ColumnMembers Property

-

Returns a **PivotColumnMembers** collection that represents all the members of the specified column.

*expression*.**ColumnMembers**

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

## Remarks

The collection of column members does not include any members that you have filtered by setting their [IsFiltered](#) property to **True**.

▼ [Show All](#)

# Columns Property

## ▶ [Column property as it applies to the \*\*Range\*\* object.](#)

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

*expression*.**Columns**

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

## ▶ [Column property as it applies to the \*\*Spreadsheet\*\* and \*\*Worksheet\*\* objects.](#)

Returns a **Range** object that represents all the columns on the specified worksheet.

*expression*.**Columns**

*expression* Required. An expression that returns one of the above objects. Read-only.

## Example

### ► [As it applies to the \*\*Range\*\* object.](#)

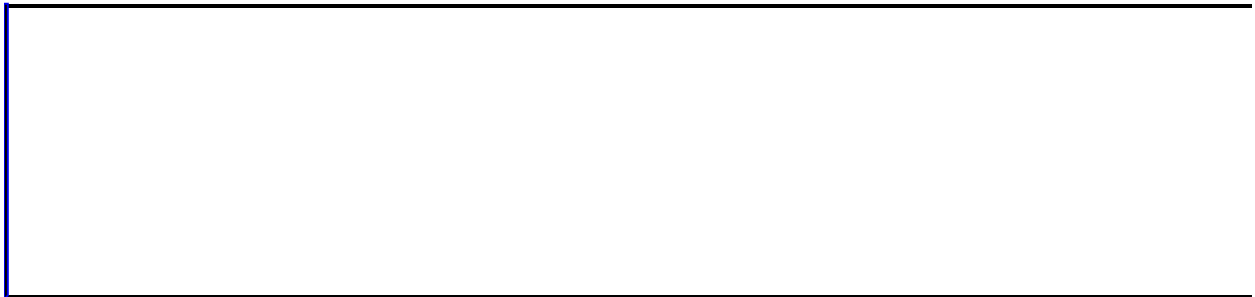
This example sets the font to bold in every other column in the visible range on the active worksheet.

```
Sub BoldColumns()  
    Dim col  
  
    For Each col In Spreadsheet1.ActiveSheet.VisibleRange.Columns  
        If col.Column Mod 2 = 0 Then col.Font.Bold = True  
    Next  
End Sub
```

### ► [As it applies to the \*\*Worksheet\*\* object.](#)

This example deletes all data from column B on the active worksheet.

```
Spreadsheet1.ActiveSheet.Columns(2).ClearContents
```





# ColumnWidth Property

-

Returns or sets the width of all columns in the specified range. Returns **Null** if the columns in the range are not all the same width. Use the **IsNull** function to determine whether the return value is **Null**. Read/write **Variant**.

*expression*.**ColumnWidth**

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

## Example

This example sets the formula, column width, and number format for all the cells in the range A1:F10 on the active worksheet.

```
Sub Format_Sheet1()  
    Dim rngCurrent  
  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Range("A1:F10")  
  
    rngCurrent.Formula = "=rand()"   
  
    rngCurrent.ColumnWidth = 10  
  
    rngCurrent.NumberFormat = "#.###"  
End Sub
```



# Commands Property

-  
Returns an **OCCommands** object that represents the collection of commands available.

*expression*.**Commands**

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

--

# CommandText Property

-

Returns or sets the command string for the specified object. Read-only **String** for the [RecordsetDef](#) and [SchemaRowsource](#) objects; read/write **String** for all other objects in the AppliesTo list.

*expression*.**CommandText**

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

## Remarks

For type **dscCommandText**, this property returns or sets a text string (usually SQL) that returns a single result set from the provider.

For type **dscCommandFile**, this property returns or sets the URL of an XML file containing data for the specified **SchemaRowsource** object.

--

▼ [Show All](#)

# CompareMemberCaptionsByProperty

Returns or sets a [PivotMembersCompareByEnum](#) constant that determines how the PivotTable control compares member captions to the list of custom captions set by the **MemberCaptions** property. Captions that match the custom list established by the **MemberCaptions** property are renamed using the custom caption. Read/write.

PivotMembersCompareByEnum can be one of these PivotMembersCompareByEnum constants.

**plMembersCompareByName** Captions are compared by name. Use this setting if you want each occurrence of a particular caption to be renamed according to the corresponding setting established by the **MemberCaptions** property. This is most useful with dates and times. For example, you may want to replace each instance of February with Feb.

**plMembersCompareByUniqueName** Captions are compared by their unique name. Use this setting when the data set contains multiple instances of a caption that are unrelated to each other. For example, the data set may contain an item named Portland under Maine and Oregon, but you do not want both of these instances to share the same caption.

*expression*.CompareMemberCaptionsBy

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

## Remarks

The default setting when the specified field set is a time-based field is **plMembersCompareByName**. Otherwise, the default setting is **plMembersCompareByUniqueName**.



▼ [Show All](#)

# CompareOrderedMembersBy Property

-

Returns or sets a [PivotMembersCompareByEnum](#) constant that determines how the PivotTable control sorts members of the specified field set when a custom sort order has been established by the **OrderedMembers** property. Read/write.

PivotMembersCompareByEnum can be one of these PivotMembersCompareByEnum constants.

**plMembersCompareByName** Captions are compared by name. Use this setting if you want each occurrence of a particular caption to be renamed according to the corresponding setting established by the **MemberCaptions** property. This is most useful with dates and times. For example, you may want to replace each instance of February with Feb.

**plMembersCompareByUniqueName** Captions are compared by their unique name. Use this setting when the data set contains multiple instances of a caption that are unrelated to each other. For example, the data set may contain an item named Portland under Maine and Oregon, but you do not want both of these instances to share the same caption.

*expression*.**CompareOrderedMembersBy**

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

## Remarks

The default setting when the specified field set is a time-based field is **plMembersCompareByName**. Otherwise, the default setting is **plMembersCompareByUniqueName**.

# Connection Property

-

Returns or sets the ADO **Connection** object used by the specified object. Read/write for the [PivotTable](#) object; read-only for the [DataSourceControl](#) and [DSCEventInfo](#) objects.

*expression*.**Connection**

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

## Remarks

Use caution when calling methods or setting properties for the **Connection** object returned by this property. For example, you should not call the **Connection** object's **Close** method.

# ConnectionString Property

Returns or sets a **String** that specifies the Office Database Connection (.odc) or Microsoft Data Link (.udl) file that is used to connect the data access page to a data source. Read/write.

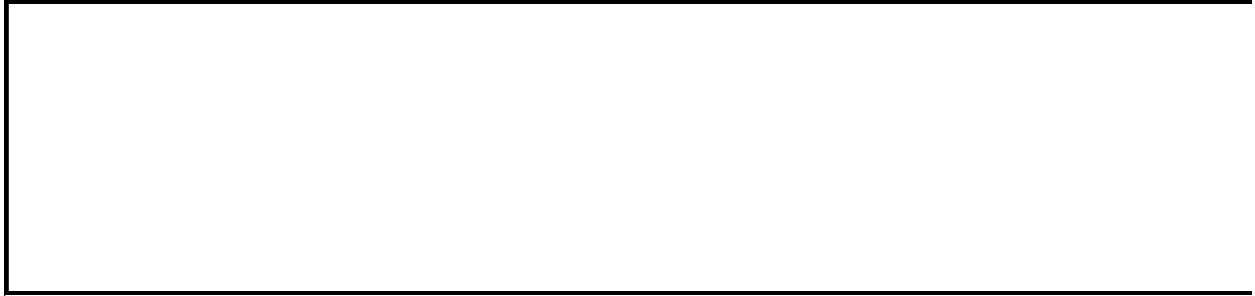
*expression*.**ConnectionString**

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

## Example

This example connects the data access page to a Office Database Connection file named "SQL Northwind.odc".

```
MSODSC.ConnectionFile = "SQL Northwind.odc"
```



# ConnectionString Property

-  
Returns or sets the ADO connection string for a two-tier database connection.  
Read/write **String**.

*expression*.**ConnectionString**

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



## Remarks

With the data source control, this property is equivalent to **CurrentProject.BaseConnectionString** in Microsoft Access. When the data source control creates a connection, the value of the data source control **ConnectionString** property may not be the same as the value returned by the **ConnectionString** property of the **Connection** object because the data source control uses other OLE DB providers to supply additional services. For example, on an HTML page containing a data source control with ID "MSODSC" the following expressions may not be equivalent.

`MSODSC.ConnectionString`

`MSODSC.Connection.ConnectionString`

## Example

This example establishes a connection to a database, queries the data, and then adds fields to PivotTable1 when the Web page containing the PivotTable1 is loaded.

```
Sub Window_OnLoad()  
  
    Dim sConnStr  
    Dim ptView  
  
    ' Set a variable to the connection string.  
    sConnStr = "provider=sqloledb;data source=mikestow01;initial cat  
  
    ' Set the connection string  
    PivotTable1.ConnectionString = sConnStr  
  
    ' Return all data from the Spending table.  
    PivotTable1.CommandText = "Select * from Spending"  
  
    Set ptView = PivotTable1.ActiveView  
  
    ' The following four lines of code add fields to the row area an  
    ' areas of the PivotTable list.  
    ptView.RowAxis.InsertFieldSet ptView.FieldSets("Project")  
    ptView.RowAxis.InsertFieldSet ptView.FieldSets("Year")  
    ptView.DataAxis.InsertFieldSet ptView.FieldSets("Budget")  
    ptView.DataAxis.InsertFieldSet ptView.FieldSets("Actual")  
  
End Sub
```



# Constants Property

-

Returns an object that allows script users to use named constants. Read-only.

For more information, see [Using Named Constants in VBScript](#).

This property is required only on HTML pages. In other containers (such as Visual Basic), you can use defined constants from the object model directly, without first using the **Constants** property. Using the **Constants** property in other containers will work but is not recommended, as it will cause your code to run significantly slower.

*expression*.**Constants**

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

## Example

This example adds a medium-weight green border to each cell in the range B5:C10 on the spreadsheet. Because named constants are not available in VBScript, the example sets a variable to the object returned by the **Constants** property.

```
Sub SetBorder()  
    Dim ssConstants  
  
    ' Set a variable to the constants of the Spreadsheet component.  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Set the border weight.  
    Spreadsheet1.Range("b5:c10").Borders.Weight = ssConstants.owcLin  
  
    ' Set the border color.  
    Spreadsheet1.Range("b5:c10").Borders.Color = "green"  
End Sub
```



# ConsumesRecordset Property

-

**True** for complex controls if data is supplied by the data source control. Any data-bound control has either a [ControlSource](#) property (used with a simple HTML control) or a **ConsumesRecordset** property (used with a complex control such as a PivotTable list or chart workspace). Read/write **Boolean**.

*expression*.**ConsumesRecordset**

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

--

# Control Property

-

Returns a **PivotTable** object that represents the PivotTable list for the specified **PivotData** or **PivotView** object.

*expression*.**Control**

*expression* Required. An expression that returns a **PivotData** or **PivotView** object.



# ControlSource Property

-  
Returns or sets the name of the control to which the specified control is bound.  
Read/write **String**.

*expression*.**ControlSource**

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

## Remarks

This property returns or sets the values of the **Name** and **Source** properties for the field to which the specified control is bound. The property values are returned as a string in the following format:

*"PageField.Name:PageField.Source"* where the strings in italics are replaced with the actual values of the **Name** and **Source** properties. If the **Name** and **Source** property values are identical, only one value is returned.

If you set this property to the name of a schema field in a schema row source used by the section containing the specified control, the schema field is automatically added as a page field.



# Count Property

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

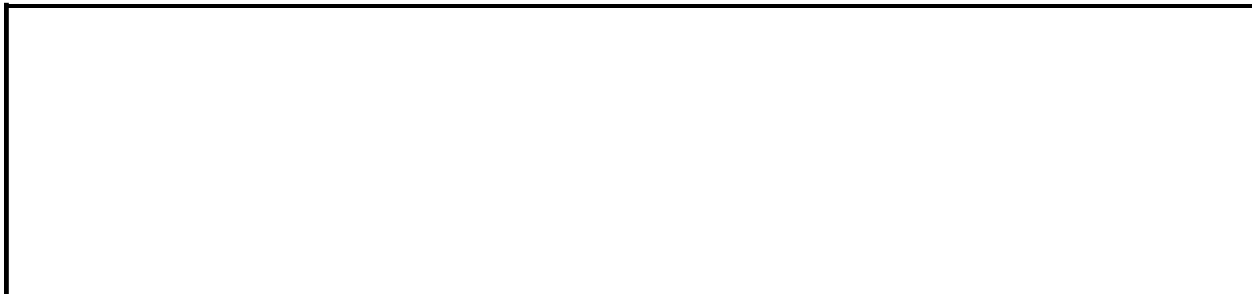
*expression*.**Count**

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

## Example

This example sets variables to the number of columns and the number of rows in the visible range on the active worksheet, and then formats the color of the cells in every other row.

```
Sub Format_Odd_Rows()  
    Dim rngUsed  
    Dim iUsedRows  
    Dim iUsedColumns  
    Dim shtActive  
    Dim iCtr  
  
    Set shtActive = Spreadsheet1.ActiveSheet  
  
    ' Set a variable of the used range of the active sheet.  
    Set rngUsed = shtActive.UsedRange  
  
    ' Get the count of used rows in the active sheet.  
    iUsedRows = rngUsed.Rows.Count  
  
    ' Get the count of used columns in the active sheet.  
    iUsedColumns = rngUsed.Columns.Count  
  
    ' Loop through every odd row in the used range.  
    For iCtr = 1 To iUsedRows Step 2  
        ' Color the background of the cells green.  
        shtActive.Range(shtActive.Cells(iCtr, 1), shtActive. _  
            Cells(iCtr, iUsedColumns)).Interior.ColorIndex = 43  
    Next  
End Sub
```



# Criteria Property

-

Returns the [Criteria](#) object for the specified filter. Use the **Add** method of the **Criteria** object returned by this property to add criteria to a **Filter** object. Read-only.

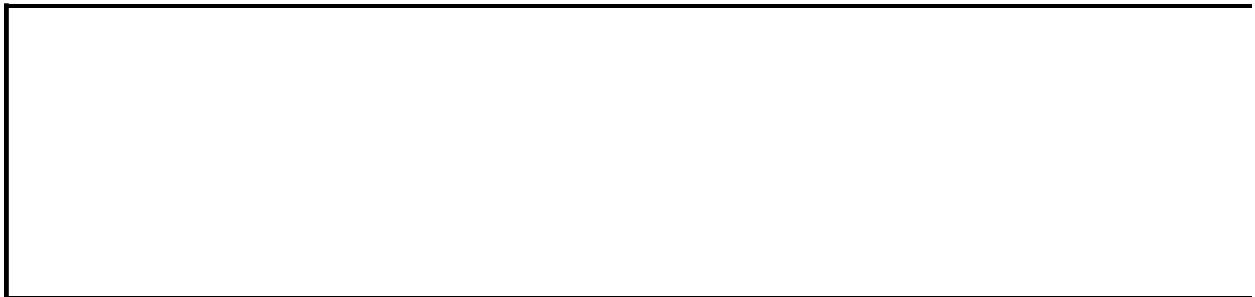
*expression*.**Criteria**

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

## Example

This example turns on the AutoFilter for the range A1:C20, sets filters for columns A and C, and then applies the filters.

```
Sub Apply_AutoFilter()  
    Dim afFilters  
    Dim afCol1  
    Dim afCol3  
  
    ' Turn on AutoFilter.  
    Spreadsheet1.Worksheets("Sheet1").Range("A1:C20").AutoFilter  
  
    ' Set a variable to the AutoFilter object  
    Set afFilters = Spreadsheet1.Worksheets("sheet1").AutoFilter  
  
    Set afCol1 = afFilters.Filters(1)  
    Set afCol3 = afFilters.Filters(3)  
  
    ' Add a criteria that excludes blue from column A.  
    afCol1.Criteria.Add "blue"  
  
    ' Add a criteria that excludes green from column A.  
    afCol1.Criteria.Add "green"  
  
    ' Add a criteria that excludes yellow from column c.  
    afCol3.Criteria.Add "yellow"  
  
    'Apply the criteria  
    afFilters.Apply  
End Sub
```



# Crosses Property

-  
Returns or sets a value that indicates how the specified axis crosses another axis.  
Read/write [ChartAxisCrossesEnum](#).

ChartAxisCrossesEnum can be one of these ChartAxisCrossesEnum constants.

**chAxisCrossesAutomatic** The crossing point for the two axes is set automatically.

**chAxisCrossesCustom** The [CrossesAtValue](#) and [CrossingAxis](#) properties specify the crossing point.

*expression*.**Crosses**

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

## Example

This example sets the category axis to cross the value axis at value zero (0) in the chart workspace if a custom crossing point has not already been set for the category axis.

```
Sub Format_Axis()  
    Dim chConstants  
    Dim axValueAxis  
    Dim axCategoryAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
    Set axCategoryAxis = ChartSpace1.Charts(0).Axes(chConstants.chAx  
  
    If axCategoryAxis.Crosses = chConstants.chAxisCrossesAutomatic T  
        axValueAxis.CrossingAxis = axCategoryAxis  
        axCategoryAxis.CrossesAtValue = 0  
    End If  
  
End Sub
```



# CrossesAtValue Property

-

Returns or sets the crossing point for the specified axis. When you set this value for an axis, you are setting the value on the *other* axis where the axis you are setting will cross that other axis. For example, setting this property on the value (y) axis sets the category number where the value axis will cross the category (x) axis. Read/write **Double**.

*expression*.**CrossesAtValue**

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

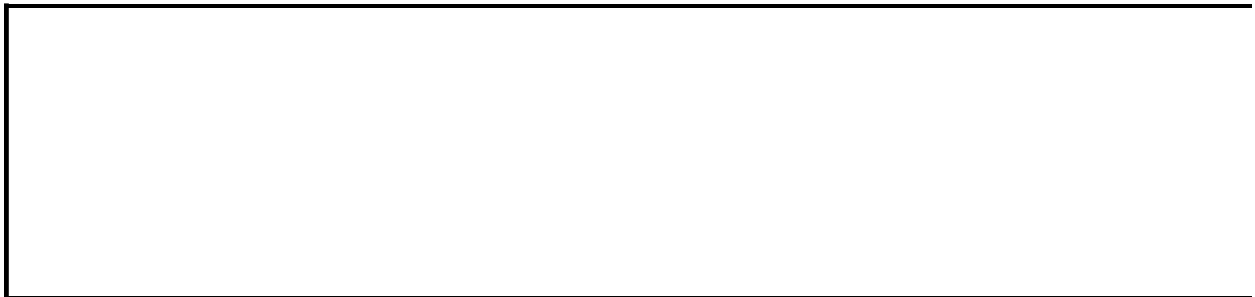
## Example

This example sets the category axis to cross the value axis at value zero (0) in the chart workspace.

```
Sub SetCrossingValue()  
    Dim chConstants  
    Dim axValueAxis  
    Dim axCategoryAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
    Set axCategoryAxis = ChartSpace1.Charts(0).Axes(chConstants.chAx  
  
    axValueAxis.CrossingAxis = axCategoryAxis  
  
    axCategoryAxis.CrossesAtValue = 0  
End Sub
```

The following example causes the value axis to cross the category axis at the third category.

```
Sub SetCrossingCategory()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
    ChartSpace1.Charts(0).Axes(chConstants.chAxisPositionLeft).Cross  
End Sub
```





# CrossingAxis Property

-

Returns or sets the crossing axis. If the specified axis is deleted, this property is set to **Null**, and the [Crosses](#) property is reset to **chAxisCrossesAutomatic**.  
Read/write **ChAxis**.

*expression*.**CrossingAxis**

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

## Example

This example sets the category axis to cross the value axis at value zero (0) in the chart workspace.

```
Sub SetCrossingValue()  
    Dim chConstants  
    Dim axValueAxis  
    Dim axCategoryAxis  
  
    Set chtContants = ChartSpace1.Constants  
  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
    Set axCategoryAxis = ChartSpace1.Charts(0).Axes(chConstants.chAx  
  
    axValueAxis.CrossingAxis = axCategoryAxis  
  
    axCategoryAxis.CrossesAtValue = 0  
End Sub
```



# CSVData Property

-  
Returns or sets spreadsheet data as a comma-delimited list. Read/write **String**.

*expression*.**CSVData**

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

## Example

This example sets the values for cells A1:G1.

Spreadsheet1.**CSVData** = "1,1,2,3,5,8,13"

1	1	2	3	5	8	13
---	---	---	---	---	---	----

# CSVURL Property

-  
Returns or sets the URL (Internet address) for the comma-delimited spreadsheet data file. Read/write **String**.

*expression*.**CSVURL**

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

## Example

This example sets spreadsheet data from a file on the user's computer.

```
Spreadsheet1.CSVURL = "file:\test.csv"
```



# CurrentArray Property

-

If the specified cell is part of an array, returns a **Range** object that represents the entire array. Results in a run-time error if the specified cell is not part of an array. Read-only.

*expression*.**CurrentArray**

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

## Remarks

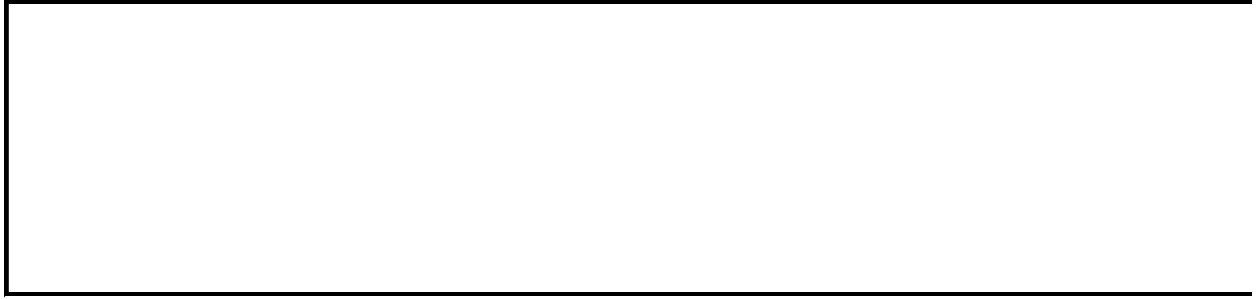
Use the **HasArray** property to determine whether or not a cell is part of an array.



## Example

Assuming that the active cell is part of an array, this example selects the array.

```
Spreadsheet1.ActiveCell.CurrentArray.Select
```



# CurrentCell Property

-

Returns a **PivotCell** object that contains the detail cell that is currently being edited.

*expression*.**CurrentCell**

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

## Remarks

This property is relevant only when the PivotTable list is connected to a relational data source.

# CurrentRegion Property

-

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.

*expression*.**CurrentRegion**

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

## Example

The function in this example returns **True**, if the entire current region for cell A1 on the active worksheet is visible (if the current region extends outside the visible range, the function returns **False**).

```
Function IsCurrentRegionVisible()  
    Dim rngCurrent  
    Dim rngVisible  
    Dim rngIntersect  
  
    ' Set the variable to the current region of cell A1.  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Cells(1, 1).CurrentRegion  
  
    ' Set a variable to the currently visible range.  
    Set rngVisible = Spreadsheet1.ActiveWindow.VisibleRange  
  
    ' Set a variable to the overlapping portion of the current region  
    ' and the visible range.  
    Set rngIntersect = Spreadsheet1.RectIntersect(rngCurrent, rngVisible)  
  
    ' If the overlapping region is the same as the current region, then  
    ' return true.  
    IsCurrentRegionVisible = (rngIntersect.Address = rngCurrent.Address)  
End Function
```



# CurrentSection Property

-

Returns a [Section](#) object that represents the current section (the section containing the control that currently has the focus).

*expression*.**CurrentSection**

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



# CustomGroupMembers Property

-

Returns a **PivotMembers** collection that represents the members of a custom group field.

*expression*.**CustomGroupMembers**

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



▼ [Show All](#)



# CustomGroupType Property

Returns a [PivotMemberCustomGroupTypeEnum](#) constant that represents the type of group that the specified member is included in. Read-only.

PivotMemberCustomGroupTypeEnum can be one of these PivotMemberCustomGroupTypeEnum constants.

**plGroupTypeCustomGroup** The specified member is a custom group.

**plGroupTypeDynamicOther**

**plGroupTypeFallThrough**

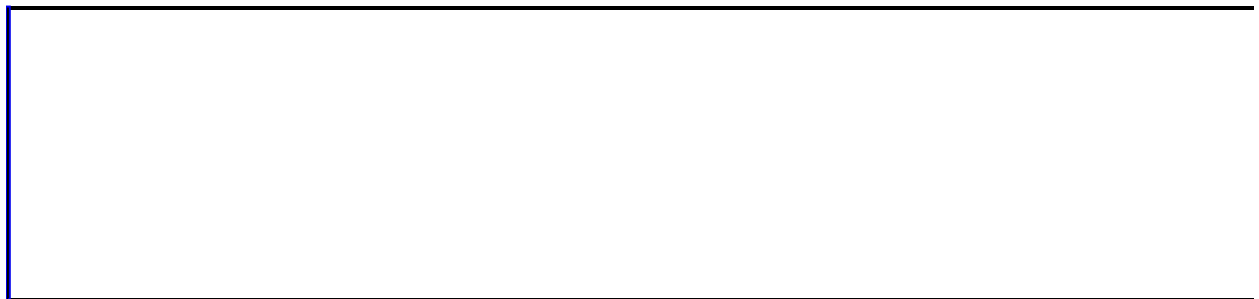
**plGroupTypePlaceholder**

**plGroupTypeRegular** The specified member is not a custom group.

**plGroupTypeStaticOther**

*expression*.CustomGroupType

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



▼ [Show All](#)

# DashStyle Property

Returns or sets a [ChartLineDashStyleEnum](#) constant indicating the dash style for the specified line or border. Read/write.

ChartLineDashStyleEnum can be one of these ChartLineDashStyleEnum constants.

**chLineDash**

**chLineDashDot**

**chLineDashDotDot**

**chLineLongDash**

**chLineLongDashDot**

**chLineRoundDot**

**chLineSolid**

**chLineSquareDot**

*expression*.**DashStyle**

*expression* Required. An expression that returns a **ChLine** or **ChBorder** object.

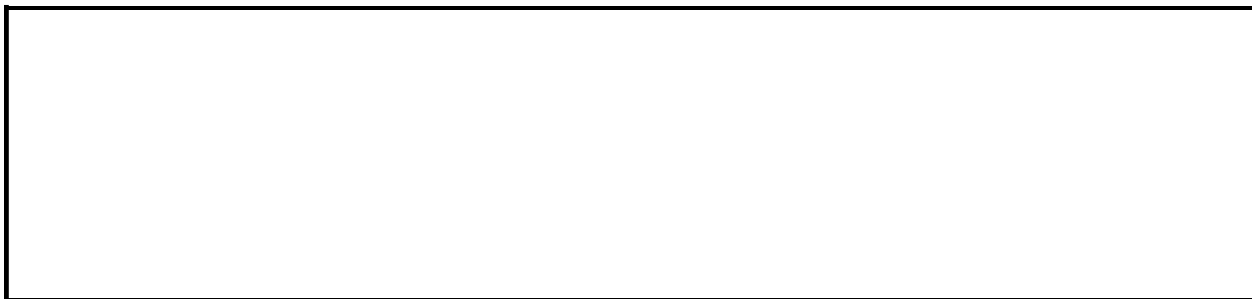
## Remarks

This property is valid only for series lines.

## Example

This example changes the first chart in Chartspace1 to a line chart and then formats the line for the first data series in the chart.

```
Sub Set_Series_LineStyle()  
  
    Dim chConstants  
    Dim serSeries1  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Change the chart to a line chart.  
    ChartSpace1.Charts(0).Type = chChartTypeLine  
  
    ' Set a variable to refer to the first data series in the chart.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Set the miter of the line of the first series.  
    serSeries1.Line.Miter = chConstants.chLineMiterBevel  
  
    ' Set the line weight of the first series.  
    serSeries1.Line.Weight = chConstants.owcLineWeightThick  
  
    ' Set the line style of the first series.  
    serSeries1.Line.DashStyle = chConstants.chLineRoundDot  
  
End Sub
```



# Data Property

-  
Returns a **PivotData** object that represents the source data for the PivotTable list.

*expression*.**Data**

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

--

# DataAxis Property

-

Returns a [PivotDataAxis](#) object that represents the data axis. Use the **PivotDataAxis** object returned by this property to insert fields and totals on the data axis of a PivotTable list. Read-only.

## Example

This example adds a total to the PivotTable list in the active view and inserts a field set on the data axis.

```
Sub InsertTotal()  
    Dim ptView  
    Dim ptConstants  
    Dim totNewTotal  
  
    Set ptView = PivotTable1.ActiveView  
    Set ptConstants = PivotTable1.Constants  
  
    Set totNewTotal = view.AddTotal("myTotal", view.FieldSets("Freig  
                                ptConstants.plFunctionSum)  
  
    ptView.DataAxis.InsertTotal totNewTotal  
    ptView.DataAxis.InsertFieldSet ptView.FieldSets("OrderDate")  
End Sub
```





# DataEntry Property

**True** if the specified page is used only for data entry (**False** if the page includes a populated recordset). The default value is **False**. Read/write **Boolean**.

*expression*.**DataEntry**

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

# DataField Property

-

Returns the name of the field that the **PivotField** object is bound to if the data is coming from a recordset. Read-only **String**.

*expression*.**DataField**

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

--

# DataLabel Property

-

Returns a **ChDataLabel** object that represents the data label associated with the specified trendline. Read-only.

*expression*.**DataLabel**

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

## Example

This example adds a trendline to the specified series in the chart workspace, sets the font for the data label to bold, and causes the trendline to display only its equation (the R-squared value is not displayed).

```
Sub AddTrendLine()  
    Dim trndline  
  
    ' Add a trendline to the first series in the first chart in  
    ChartSpace1.  
    Set trndline =  
    ChartSpace1.Charts(0).SeriesCollection(0).Trendlines.Add  
  
    ' Set the font of the trendline to bold.  
    trndline.DataLabel.Font.Bold = True  
  
    ' Do not display the R-Squared value with the trendline.  
    trndline.IsDisplayingRSquared = False  
  
    ' Display the equation for the trendline.  
    trndline.IsDisplayingEquation = True  
End Sub
```



# DataLabelsCollection Property

-

Returns a **ChDataLabelsCollection** object that contains the data labels for the specified series. Each series can contain only one set of data labels. Read-only.

For more information about returning a single member of a collection, see [Returning an Object from a Collection](#).

*expression*.**DataLabelsCollection**

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

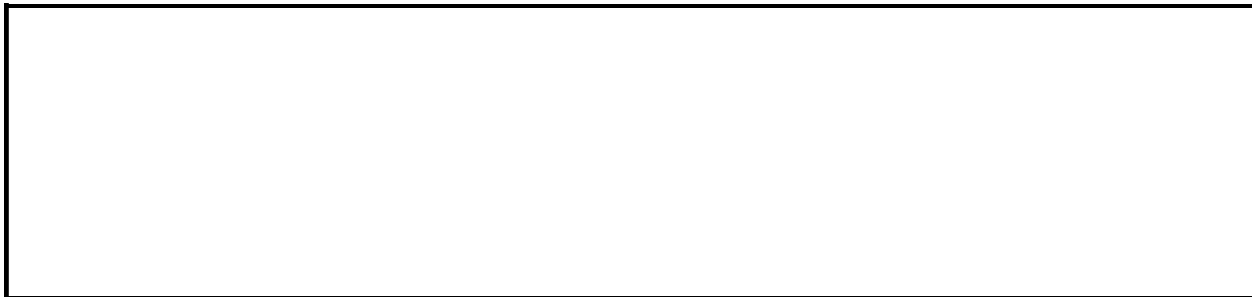
## Example

This example adds data labels to the specified series in the chart workspace.

```
ChartSpace1.Charts(0).SeriesCollection(0).DataLabelsCollection.Add
```

This example sets the font for the data labels for the specified series.

```
ChartSpace1.Charts(0).SeriesCollection(0).DataLabelsCollection(0).Fo
```



# DataMember Property

-  
Returns or sets the data member name (the name of the recordset that the specified control will request from the data source). Read/write **DataMember**.

*expression*.**DataMember**

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

--

# DataPage Property

-  
Returns a [DataPage](#) object that represents the data access page for the specified section.

*expression*.**DataPage**

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



## Remarks

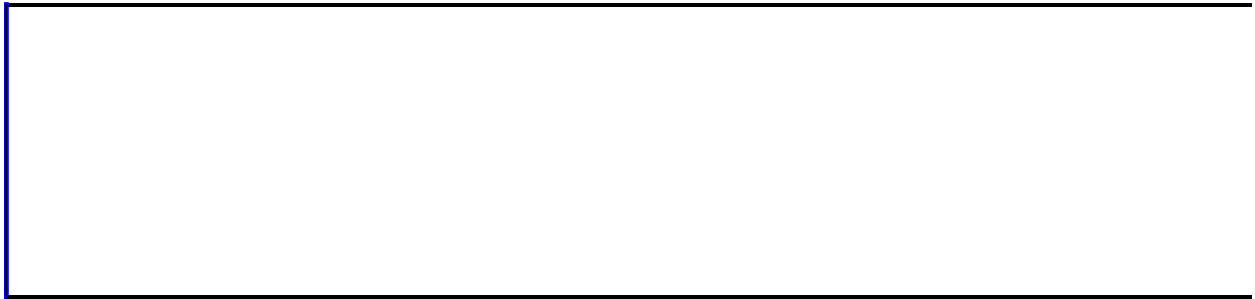
A data page is a cluster of sections that share a common record navigation control.

# DataPages Property

-  
Returns the [DataPages](#) collection for the data source control.

*expression*.**DataPages**

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



# DataPageSize Property

-

Returns or sets the number of records shown for the specified banding level on a data access page. The default value is 5 for a banded page and 1 for a non-banded page. Read/write **Long**.

*expression*.**DataPageSize**

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

---

---

# DataSource Property

-  
Returns or sets the ADO **DataSource** object that represents the data source for the specified control.

*expression*.**DataSource**

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

## Example

This example creates a chart that is bound to a spreadsheet. The series name is in cell B1, the category names are in cells A2:A28, and the values are in cells B2:B28.

```
Set c = ChartSpace1.Constants
Set ChartSpace1.DataSource = Spreadsheet1.Object
ChartSpace1.Charts.Add
ChartSpace1.Charts(0).Type = c.chChartTypeLineMarkers
ChartSpace1.Charts(0).SetData c.chDimCategories, 0, "a2:a28"
ChartSpace1.Charts(0).SetData c.chDimSeriesNames, 0, "b1"
ChartSpace1.Charts(0).SeriesCollection(0).SetData c.chDimValues, 0,
```



# DataSourceName Property

-

Returns or sets a **String** specifying the name of the ActiveX control that serves as the data source for the chart control. This property can be used in containers that support the Microsoft Internet Explorer Document Object Model.

Read/write.

*expression*.**DataSourceName**

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

## Example

This example establishes PivotTable1 as the data source for ChartSpace1 and then displays the field list so that the user can add data to the chart.

```
Sub ConnectChart()  
    ' Bind ChartSpace1 to PivotTable1.  
    ChartSpace1.DataSourceName = "PivotTable1"  
  
    ' Display the field list.  
    ChartSpace1.DisplayFieldList = True  
End Sub
```



# DataSourceType Property

-

Returns or sets the data source type for the specified chart workspace. Read-only [ChartDataSourceTypeEnum](#).

ChartDataSourceTypeEnum can be one of these ChartDataSourceTypeEnum constants.

**chDataSourceTypeDSC**

**chDataSourceTypePivotTable**

**chDataSourceTypeQuery**

**chDataSourceTypeSpreadsheet**

**chDataSourceTypeUnknown**

---

*expression*.**DataSourceType**

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



▼ [Show All](#)

# Data Type Property

▸ [DataType property as it applies to the \*\*SchemaField\*\* and \*\*SchemaParameter\*\* objects.](#)

Returns or sets the specified field's data type. Read/write ADO **DataTypeEnum**.

*expression*.**DataType**

*expression* Required. An expression that returns one of the above objects.

▸ [DataType property as it applies to the \*\*PivotField\*\* object.](#)

Returns the specified field's data type. Read-only ADO **DataTypeEnum**.

*expression*.**DataType**

*expression* Required. An expression that returns a [PivotField](#) object.

▸ [DataType property as it applies to the \*\*Spreadsheet\*\* object.](#)

Specifies the spreadsheet's data format and source, as shown in the following table. Read/write **String**.

Value	Meaning
HTMLURL	The data source is the URL specified by the <a href="#">HTMLURL</a> property.
HTMLData	The data source is the string specified by the <a href="#">HTMLData</a> property.
CSVURL	The data source is the URL specified by the <a href="#">CSVURL</a> property.
CSVData	The data source is the string specified by the <a href="#">CSVData</a> property.

*expression*.**DataType**

*expression* Required. An expression that returns a [Spreadsheet](#) object.

--

# DefaultColor Property

-

Returns a RGB-packed integer that represents the specified fill color when the **Color** property is **chColorAutomatic**. Read-only **Variant**.

*expression*.**DefaultColor**

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



# DefaultControlType Property

-

Returns or sets the default control type for the specified data source control.  
Read/write [DefaultControlTypeEnum](#).

DefaultControlTypeEnum can be one of these DefaultControlTypeEnum constants.

**ctlTypeBoundHTML**

**ctlTypeBoundSpan**

**ctlTypeTextBox**

---

*expression*.**DefaultControlType**

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

# DefaultMember Property

-

Returns a **PivotMember** object that represents the default member for the specified field set.

*expression*.**DefaultMember**

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

## Remarks

This property is valid only when the PivotTable list is connected to an OLAP data source.

The default member is usually *All* or one of the top members in the dimension.

# DefaultRecordset Property

Returns the default ADO **Recordset** object for the specified control. Read-only.

*expression*.**DefaultRecordset**

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



## Remarks

You should not use this property with sections that are bound to recordsets; instead, use the **DataPage** object's [Recordset](#) property.

# DefaultSort Property

-

Returns or sets the field (or fields) on which the specified banding level is sorted. The field must be part of the banding level. If you use this property to specify multiple fields, use commas to separate the fields. Read/write **String**.

*expression*.**DefaultSort**

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

---

---

# DefaultValue Property

-

Returns or sets the value that is placed in the specified control whenever the user creates a new record. Applies to any bound control, any control with a **Value** property, or any control that meets both of these criteria. Read/write **String** for the [ElementExtension](#) object; read/write **Variant** for the [PivotField](#) object.

*expression*.**DefaultValue**

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

--

# DesignMode Property

**True** if the spreadsheet control is currently in design mode. Read-only **Boolean**.

*expression*.**DesignMode**

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

---

---

▼ [Show All](#)

# DetailAutoFit Property

▸ [DetailAutoFit property as it applies to the \*\*PivotField\*\* object.](#)

**True** if the column width for the specified field is set automatically when the field is used on the data axis. Read/write **Boolean**.

*expression*.**DetailAutoFit**

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

▸ [DetailAutoFit property as it applies to the \*\*PivotView\*\* object.](#)

**True** if the width of the detail grid is determined by the width of the detail fields. The width of the detail grid cannot exceed the value of the [DetailMaxWidth](#) property. If the **DetailAutoFit** property is set to **False**, the width of the detail grid is set based on the value of the [DetailWidth](#) property. If the **DetailAutoFit** property is set to **True**, any layout change automatically updates the **DetailWidth** property. Read/write **Boolean**.

---

*expression*.**DetailAutoFit**

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

# DetailBackColor Property

-

Returns or sets the back color for the specified field when the field is displayed in a detail grid. The default back color is white. Read/write **Variant**.

*expression*.**DetailBackColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---



# DetailCells Property

-

Returns a [PivotDetailCell](#) object based on a row and column index. Use this property to access the properties of individual cells in the detail area of a PivotTable list.

*expression*.**Cell**(*Row*, *Column*)

*expression* An expression that returns a [PivotCell](#) object.

**Row** Required **Long**. Specifies the row that contains the specified cell.

**Column** Required **Long**. Specifies the column that contains the specified cell.



# DetailColumnCount Property

Returns a **Long** value that represents the number of columns of detail cells in the specified cell. Read-only.

*expression*.**DetailColumnCount**

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

## Remarks

Using this property when the PivotTable list is connected to an OLAP data source will result in a run-time error.

# DetailFont Property

-

Returns a [PivotFont](#) object that represents the font used when the specified field is displayed in the detail grid. Read-only.

---

*expression*.**DetailFont**

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

# DetailForeColor Property

-

Returns or sets the foreground color for the specified field when the field is displayed in a detail grid. Read/write **Variant**.

*expression*.**DetailForeColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---

## Example

This example adds inserts fields into PivotTable1, add a total, and then formats the field in the detail area of the PivotTable list.

```
Sub Layout_PivotTable1()  
    Dim vwView  
    Dim ptConstants  
    Dim totOrderCount  
  
    Set ptConstants = PivotTable1.Constants  
    Set vwView = PivotTable1.ActiveView  
  
    ' Add the ShipCountry field to the row axis.  
    vwView.RowAxis.InsertFieldSet vwView.FieldSets("ShipCountry")  
  
    ' Add the OrderId field to the data axis.  
    vwView.DataAxis.InsertFieldSet vwView.FieldSets("OrderID")  
  
    ' Add the ShipVia field to the filter axis.  
    vwView.FilterAxis.InsertFieldSet vwView.FieldSets("ShipVia")  
  
    ' Create a total named "Order Count" that counts the OrderID fie  
    Set totOrderCount = vwView.AddTotal("Order Count", vwView.Fields  
        ptConstants.plFunctionCount)  
  
    ' Add the Order Count total to the data axis.  
    vwView.DataAxis.InsertTotal totOrderCount  
  
    ' Set the horizontal alignment of the OrderID field.  
    vwView.FieldSets("OrderId").Fields("OrderId").DetailHAlignment =  
  
    ' Set the foreground color of the OrderId field.  
    vwView.FieldSets("OrderId").Fields("OrderId").DetailForeColor =  
  
End Sub
```

# DetailHAlignment Property

Returns or sets the way field values are horizontally aligned when the specified field is displayed in a detail grid. By default, values are left-aligned. Read/write [PivotHAlignmentEnum](#).

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

---

*expression*.**DetailHAlignment**

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



## Example

This example adds inserts fields into PivotTable1, add a total, and then formats the field in the detail area of the PivotTable list.

```
Sub Layout_PivotTable1()  
    Dim vwView  
    Dim ptConstants  
    Dim totOrderCount  
  
    Set ptConstants = PivotTable1.Constants  
    Set vwView = PivotTable1.ActiveView  
  
    ' Add the ShipCountry field to the row axis.  
    vwView.RowAxis.InsertFieldSet  
    vwView.FieldSets("ShipCountry")  
  
    ' Add the OrderId field to the data axis.  
    vwView.DataAxis.InsertFieldSet vwView.FieldSets("OrderID")  
  
    ' Add the ShipVia field to the filter axis.  
    vwView.FilterAxis.InsertFieldSet  
    vwView.FieldSets("ShipVia")  
  
    ' Create a total named "Order Count" that counts  
    ' the OrderID field.  
    Set totOrderCount = vwView.AddTotal("Order Count",  
    vwView.FieldSets("OrderId").Fields("OrderId"), _  
    ptConstants.plFunctionCount)  
  
    ' Add the Order Count total to the data axis.  
    vwView.DataAxis.InsertTotal totOrderCount  
  
    ' Set the horizontal alignment of the OrderID field.  
    vwView.FieldSets("OrderId").Fields("OrderId").DetailHAlignment =  
  
    ' Set the foreground color of the OrderId field.  
    vwView.FieldSets("OrderId").Fields("OrderId").DetailForeColor =  
  
End Sub
```

# DetailLeft Property

-

Returns or sets a **Long** that represents the leftmost visible column in the detail grid. Read/write.

*expression*.**DetailLeft**(*Column*)

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

**Column** Required [PivotMember](#) object. Specifies the leftmost visible column in the detail grid.



# DetailLeftOffset Property

-

Returns or sets a **Long** value that represents the number of pixels that the data in the detail area is offset to the left. Use this property to move the data in the detail area to the left. Read/write.

*expression*.**DetailLeftOffset**

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

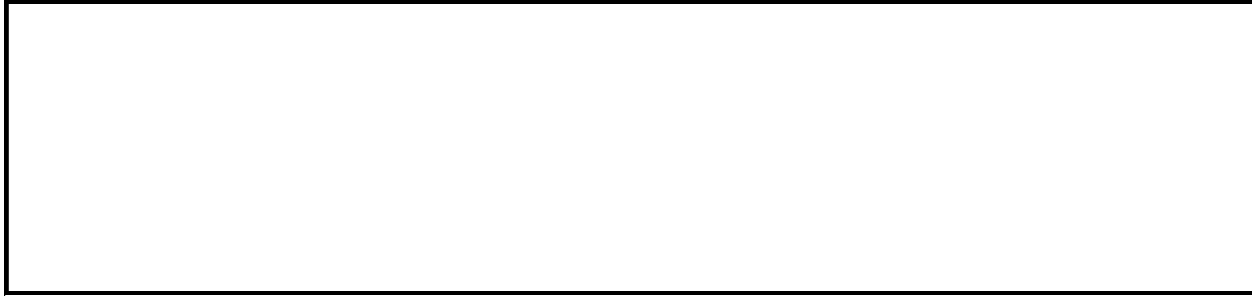
## Remarks

Setting this property to a negative value will result in a run-time error.

## Example

This example moves the data in the detail area of PivotTable1 10 pixels to the left.

```
PivotTable1.ActiveData.ColumnAxis.ColumnMember.DetailLeftOffset = 10
```



# DetailMaxHeight Property

-

Returns or sets the maximum height (in pixels) that the detail grid can attain when the value of the [DetailAutoFit](#) property is **True**. A scroll bar is displayed if the height of the detail grid would exceed the maximum height. The default value is 250 pixels. This property is ignored if the value of the **DetailAutoFit** property is **False**. Read/write **Long**.

---

*expression*.**DetailMaxHeight**

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

# DetailMaxWidth Property

-

Returns or sets the maximum width (in pixels) that the detail grid can attain when the value of the [DetailAutoFit](#) property is **True**. A scroll bar is displayed if the width of the detail grid would exceed the maximum width. The default value is 1024 pixels. This property is ignored if the value of the **DetailAutoFit** property is **False**. Read/write **Long**.

---

*expression*.**DetailMaxWidth**

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

# DetailRange Property

-  
Returns a [PivotDetailRange](#) object for the specified area.

*expression*.**DetailRange**(*TopLeft*, *BottomRight*)

*expression* An expression that returns a [PivotCell](#) object.

**TopLeft** Required [PivotDetailCell](#) object. Specifies the upper-left cell in the detail range.

**BottomRight** Required **PivotDetailCell** object. Specifies the lower-right cell in the detail range.





# DetailRowCount Property

Returns a **Long** value that represents the number of rows in the detail area that contains the specified cell. Read-only.

*expression*.**DetailRowCount**

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

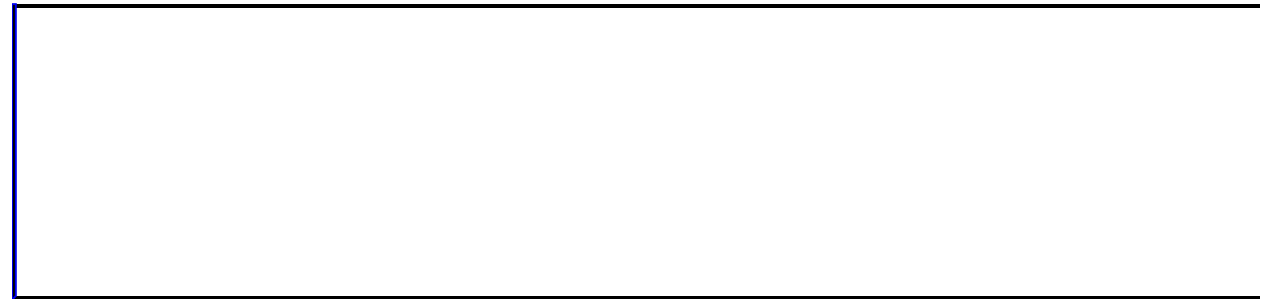
## Remarks

Using this property when the PivotTable list is connected to an OLAP data source will result in a run-time error.

# DetailRowHeight Property

-

Returns or sets the row height for the detail grid (in pixels). The default value is 10 pixels. Read/write **Long**.



*expression*.**DetailRowHeight**

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

# DetailsExpanded Property

-

Returns whether or not the details have been expanded in the detail area of the PivotTable list. Read-only **Boolean**.

*expression*.**DetailsExpanded**

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

## Remarks

Using this property when the PivotTable list is connected to an OLAP data source will result in a run-time error.

# DetailSortOrder Property

-  
Returns or sets the sort order for fields on the summary axis. Read/write **Variant**.

*expression*.**DetailSortOrder**

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

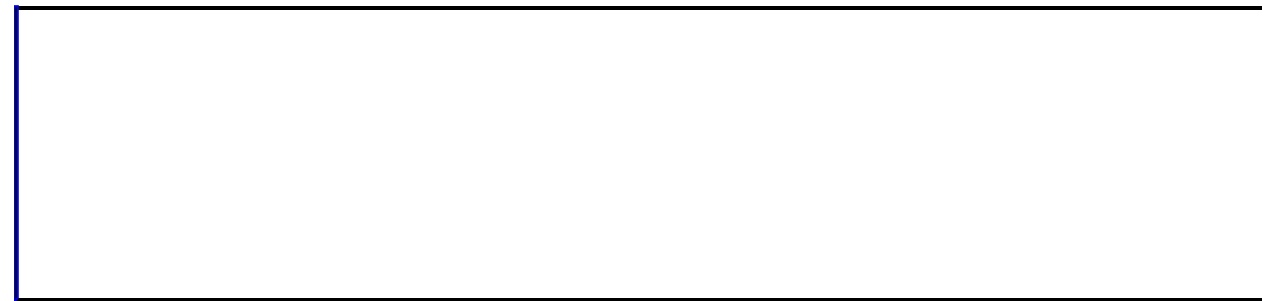
## Remarks

Fields are sorted in array order. If a field included in the sort order array has its [SortDirection](#) property set to **plSortDirectionDefault**, the field is ignored.

# DetailTop Property

-

Returns or sets the index of the uppermost visible row in the detail grid.  
Read/write **Long**.



*expression*.**DetailTop**

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



# DetailTopOffset Property

-

Returns or sets a **Long** value that represents the number of pixels that the specified cell in the detail area is offset to the top. Use this property to move a cell in the detail area upward. Read/write.

*expression*.**DetailTopOffset**

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

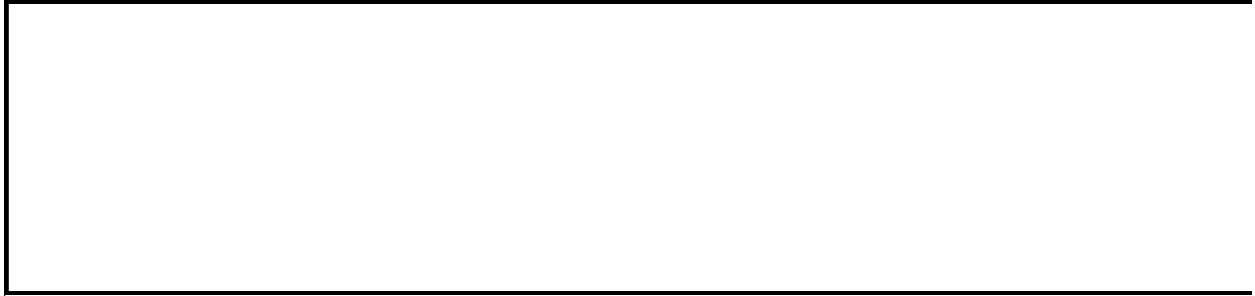
## Remarks

Setting this property to a negative value will result in a run-time error.

## Example

This example moves scrolls the currently selected cell in the detail area of PivotTable1 up by 15 pixels

```
PivotTable1.ActiveData.CurrentCell.DetailTopOffset = 15
```



# DetailWidth Property

Returns or sets the width of the specified field (in pixels) when it is displayed in the detail grid. Read/write **Long**.

*expression*.**DetailWidth**

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



▼ [Show All](#)

# Direction Property

▸ [Direction property as it applies to the \*\*ChErrorBars\*\* object.](#)

Returns or sets the error-bar direction for the specified error bars. Read/write [ChartErrorBarDirectionEnum](#).

ChartErrorBarDirectionEnum can be one of these ChartErrorBarDirectionEnum constants.

**chErrorBarDirectionX**

**chErrorBarDirectionY**

*expression*.**Direction**

*expression* Required. An expression that returns one of the above objects.

## Remarks

You can use the **chErrorBarDirectionX** constant only in xy (scatter), polar, bar, or bubble charts.

► [Direction property as it applies to the SchemaParameter object.](#)

Returns the **Direction** parameter for the [SchemaRowSource](#) object. Read-only **ParameterDirectionEnum** (ADO).

*expression*.**Direction**

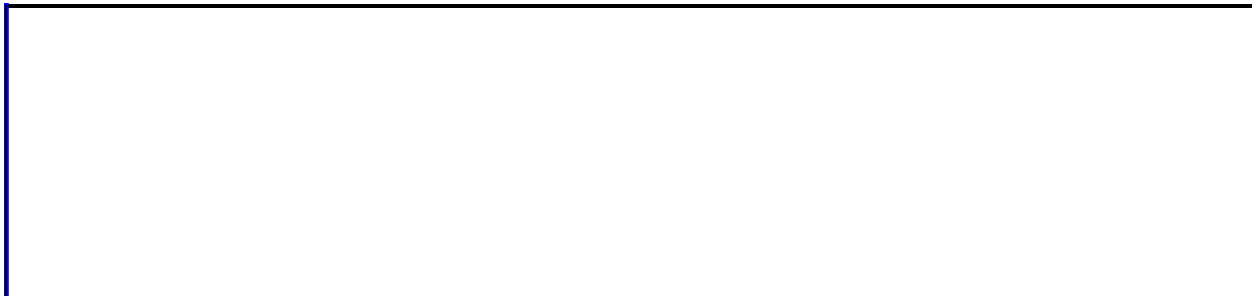
*expression* Required. An expression that returns one of the above objects.

## Example

► [As it applies to the \*\*ChErrorBars\*\* object.](#)

This example adds error bars to the specified chart and sets the error-bar direction. Charts(0) must refer to an xy (scatter), polar, bar, or bubble chart.

```
Sub AddErrorBars()  
  
    Dim ebErrorBars  
    Dim chConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Add error bars to the first series in the chart.  
    Set ebErrorBars = ChartSpace1.Charts(0).SeriesCollection(0).Error  
  
    ' Set the direction of the error bars.  
    ebErrorBars.Direction = chConstants.chErrorBarDirectionX  
  
End Sub
```





# DirectionalLightInclination Property

-

Returns or sets a **Double** specifying the rotation of the directional light source along the x-z plane of the specified chart. Valid settings range from -90 to 90. Setting this property to -90 places the light source directly below the chart. Setting this property to 90 places the light source directly above the chart. Read/write.

*expression*.**DirectionalLightInclination**

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

## Example

This example converts the first chart in ChartSpace1 to a 3-D Bar chart and sets the lighting options for the chart.

```
Sub Format3DLightSources()  
  
    Dim cht3DBar  
  
    ' Set a variable to the first chart in ChartSpace1.  
    Set cht3DBar = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Bar chart.  
    cht3DBar.Type = chChartTypeBar3D  
  
    ' Set the intensity of the ambient light.  
    cht3DBar.AmbientLightIntensity = 0.7  
  
    ' Set the inclination of the directional light source.  
    cht3DBar.DirectionalLightInclination = 35  
  
    ' Set the intensity of the directional light source.  
    cht3DBar.DirectionallLightIntensity = 0.8  
  
    ' Set the rotation of the directional light source.  
    cht3DBar.DirectionallLightRotation = 120  
  
End Sub
```



# DirectionalLightIntensity Property

-

Returns or sets a **Double** specifying the intensity of the directional light source for a three-dimensional chart. Valid settings range from 0 to 1. Read/write.

*expression*.**DirectionalLightIntensity**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Bar chart and sets the lighting options for the chart.

```
Sub Format_3D_LightSources()  
  
    Dim cht3DBar  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DBar = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Bar chart.  
    cht3DBar.Type = chChartTypeBar3D  
  
    ' Set the intensity of the ambient light.  
    cht3DBar.AmbientLightIntensity = 0.7  
  
    ' Set the inclination of the directional light source.  
    cht3DBar.DirectionallLightInclination = 35  
  
    ' Set the intensity of the directional light source.  
    cht3DBar.DirectionallLightIntensity = 0.8  
  
    ' Set the rotation of the directional light source.  
    cht3DBar.DirectionallLightRotation = 120  
End Sub
```



# DirectionalLightRotation Property

-

Returns or sets a **Double** specifying the rotation of the directional light source for a three-dimensional chart. Valid settings are between 0 and 360. Read/write.

*expression*.**DirectionalLightRotation**

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

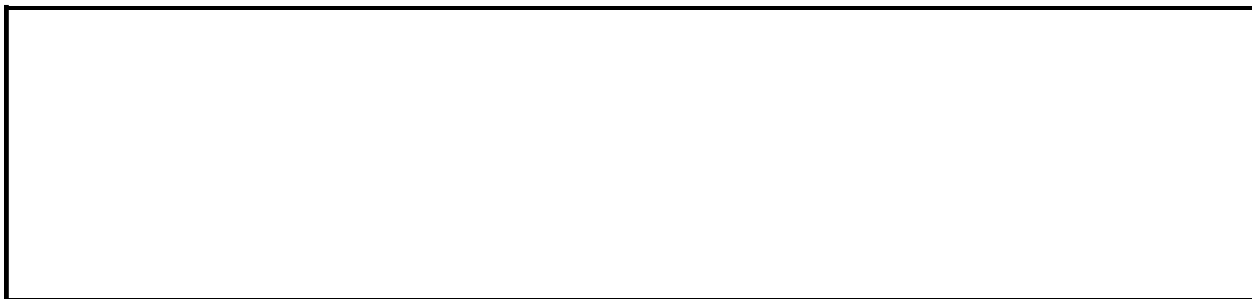
## Remarks

The directional light source rotates around the y-axis of the chart.

## Example

This example converts the first chart in Chartspace1 to a 3-D Bar chart and sets the lighting options for the chart.

```
Sub Format3DLightSources()  
  
    Dim cht3DBar  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DBar = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Bar chart.  
    cht3DBar.Type = chChartTypeBar3D  
  
    ' Set the intensity of the ambient light.  
    cht3DBar.AmbientLightIntensity = 0.7  
  
    ' Set the inclination of the directional light source.  
    cht3DBar.DirectionallightInclination = 35  
  
    ' Set the intensity of the directional light source.  
    cht3DBar.DirectionallightIntensity = 0.8  
  
    ' Set the rotation of the directional light source.  
    cht3DBar.DirectionallightRotation = 120  
  
End Sub
```



# Dirty Property

**True** if changes have been made to the spreadsheet since the last time it was saved. Read/write **Boolean**.

*expression*.**Dirty**

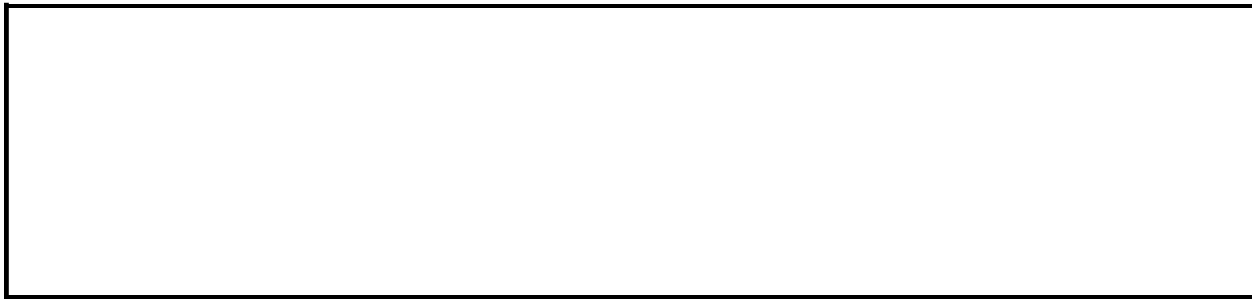
*expression* Required. An expression that returns a **Spreadsheet** object.



## Example

This example exports the spreadsheet to a file if changes have been made to it since the last time it was saved.

```
If Spreadsheet1.Dirty Then  
    Spreadsheet1.Export "sstest.xls", Spreadsheet1.Constants.ssExpor  
End If
```



▼ [Show All](#)

# DisplayAlert Property

-

Returns or sets a [DscDisplayAlert](#) constant that indicates whether or not an alert will be displayed when the BeforeDelete and BeforeOverwrite events are called. Read/write.

DscDisplayAlert can be one of these DscDisplayAlert constants.

**dscDataAlertContinue** An alert is not displayed.

**dscDataAlertDisplay** An alert is displayed asking the user to confirm the action.

*expression*.**DisplayAlert**

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

## Example

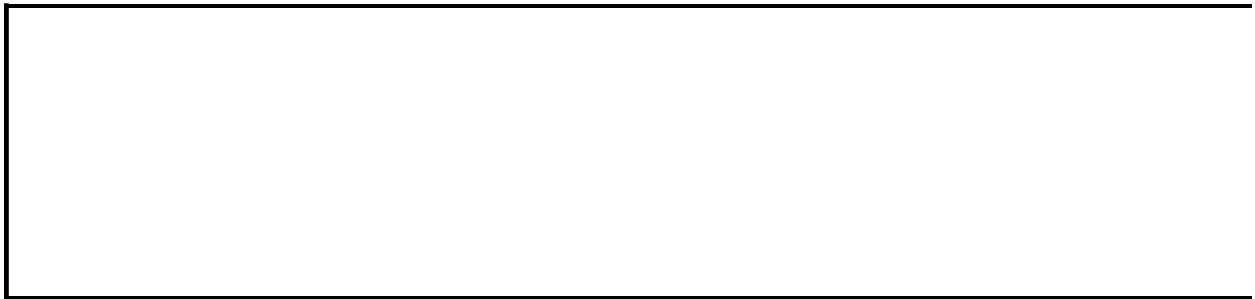
The following example uses the **DisplayAlert** property in the **BeforeOverwrite** event to prevent the user from being prompted to overwrite an existing file when the **ExportXML** method is called.

```
Sub MSODSC_BeforeOverwrite(DSCEventInfo)

    Dim dscConstants
    Set dscConstants = MSODSC.Constants

    ' Don't alert the user when overwriting an existing file.
    DSCEventInfo.DisplayAlert = dscConstants.dscDataAlertContinue

End Sub
```



# DisplayAlerts Property

**True** if certain alerts and messages are to be displayed while code is running. The default value is **True**. Set this property to **False** if you don't want to be disturbed by prompts and alert messages; any time a message requires a response, the default response is chosen.

If you set this property to **False**, it is not automatically set back to **True**, and must be set to **True** for messages and alerts to appear. Read/write **Boolean**.

*expression*.**DisplayAlerts**

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

--

# DisplayCalculatedMembers Property

-

Returns or sets a **Boolean** that determines whether calculated members are displayed. Set this property to **False** to hide calculated members. The default value is **True**. Read/write.

*expression*.**DisplayCalculatedMembers**

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

---

---

# DisplayColumnHeadings Property

-

Returns or sets whether column headings are displayed in the specified window. Set this property to **False** to hide the columns headings. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayColumnHeadings**

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

## Example

This example hides the row and column headings in the active window of Spreadsheet1.

```
Sub HideHeadings()  
    Spreadsheet1.ActiveWindow.DisplayColumnHeadings = False  
    Spreadsheet1.ActiveWindow.DisplayRowHeadings = False  
End Sub
```





# DisplayCustomHeadings Property

-

Determines whether custom row and column headings, if they exist, are to be displayed in the specified window. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayCustomHeadings**

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

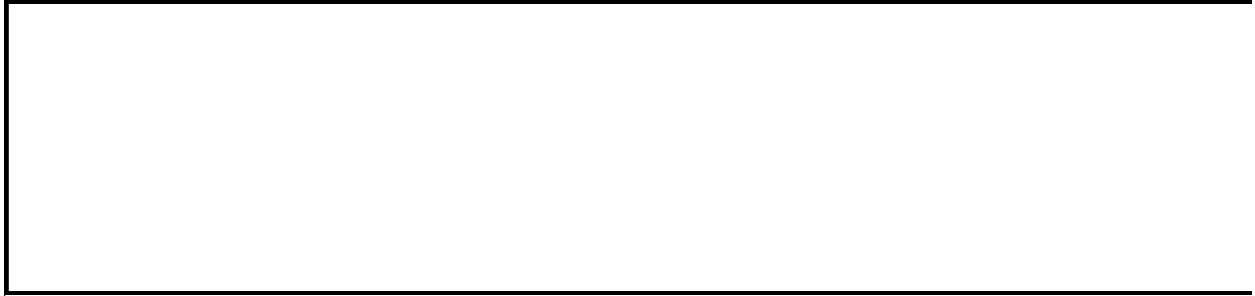
## Remarks

Although setting this property to **False** hides custom row and column headings, they are not reset them to their default values. Use this property instead of using the **ResetHeadings** method to temporarily hide the custom row and column headings.

## Example

This example prevents the display of the custom row and column headings in the active window of Spreadsheet1.

```
Spreadsheet1.ActiveWindow.DisplayCustomHeadings = False
```



# DisplayDesignTimeUI Property

**True** to display the design-time version of the Commands and Options window is displayed at run time. The default value is **False**. Read/write **Boolean**.

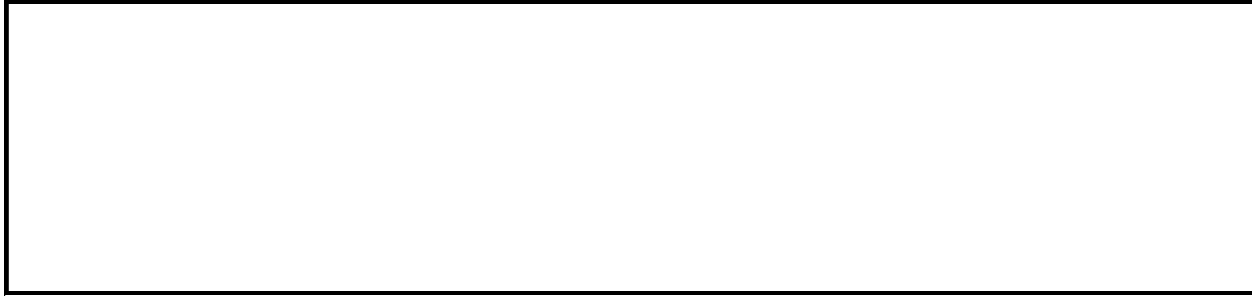
*expression*.**DisplayDesignTimeUI**

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

## Example

This example sets Spreadsheet1 so that the design-time Commands and Options window will be displayed at run time.

```
Spreadsheet1.DisplayDesignTimeUI = True
```



# DisplayEmptyMembers Property

-

**True** if empty members are displayed, even if they do not have aggregates. The default value is **False**. Read/write **Boolean**.

---

*expression*.**DisplayEmptyMembers**

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

# DisplayExpandIndicator Property

-

**True** if expansion indicators are displayed for members with available child members or detail records. When the expansion indicator is hidden, the member display name appears in this space. The default value is **True**. Read/write **Boolean**.

---

*expression*.**DisplayExpandIndicator**

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

# DisplayFieldButtons Property

-

Returns or sets whether field buttons and drop zones are displayed on the chart. Set this property to **False** to hide the field buttons and drop zones. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayFieldButtons**

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



## Remarks

This property has no effect if the chart control is bound to a literal data source.

## Example

This example hides the drop zones and field buttons on Chartspace1.

```
Chartspace1.DisplayFieldButtons = False
```



# DisplayFieldList Property

-

Set this property to **True** to display the Field List. The default value is **False**.  
Read/write **Boolean**.

*expression*.**DisplayFieldList**

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

# DisplayGridlines Property

**True** if gridlines are displayed on the specified spreadsheet. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayGridlines**

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

## Example

This example hides gridlines on the spreadsheet.

```
Spreadsheet1.ActiveWindow.DisplayGridlines = False
```



# DisplayHeadings Property

-  
**True** if both row and column headings are displayed; **False** if there are no headings displayed, or if either the column or row headings are not displayed.  
Read/write **Boolean**.

*expression*.**DisplayHeadings**

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

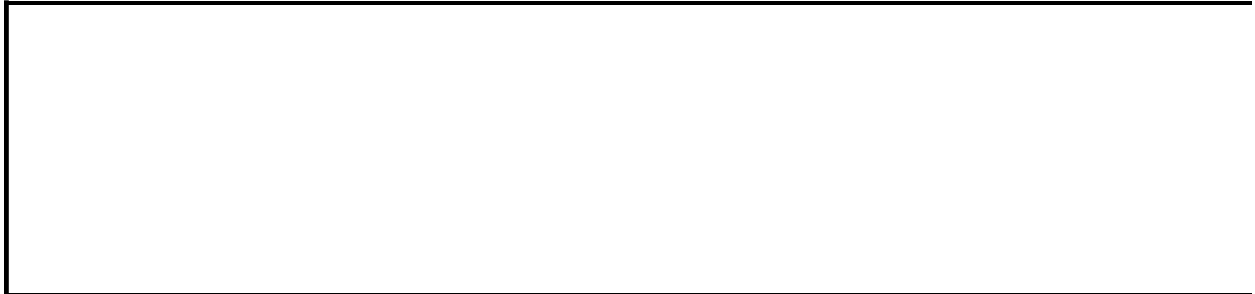
## Remarks

You can use the **DisplayColumnHeadings** and **DisplayRowHeadings** properties to independently control the display of column and row headings.

## Example

This example hides the row and column headings in the active window of Spreadsheet1.

```
Spreadsheet1.ActiveWindow.DisplayHeadings = False
```





# DisplayHorizontalScrollBar Property

**True** if the horizontal scroll bar is displayed on the specified spreadsheet. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayHorizontalScrollBar**

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

## Example

This example hides the horizontal scroll bar on the spreadsheet.

```
Spreadsheet1.ActiveWindow.DisplayHorizontalScrollBar = False
```



▼ [Show All](#)

# DisplayIn Property

-

Returns or sets a [PivotMemberPropertyDisplayEnum](#) constant that determines whether the specified member property is displayed in the PivotTable list, ScreenTip, both the PivotTable list and ScreenTip, or not at all. Read/write.

PivotMemberPropertyDisplayEnum can be one of these PivotMemberPropertyDisplayEnum constants.

**plDisplayPropertyInAll**

**plDisplayPropertyInReport**

**plDisplayPropertyInScreenTip**

**plDisplayPropertyNone**

*expression*.**DisplayIn**

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

## Example

This example sets the captions of, and then displays the member captions of the Store Name field.

```
Sub Display_MemberProperties()  
    Dim ptView  
    Dim ptConstants  
    Dim fldStoreName  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view of the PivotTable.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Store Name field.  
    Set fldStoreName = ptView.FieldSets("Store").Fields("Store Name")  
  
    ' The following three lines of code specify that the member prop  
    ' displayed in the PivotTable list.  
    fldStoreName.MemberProperties("Store Manager").DisplayIn = ptCon  
    fldStoreName.MemberProperties("Store Type").DisplayIn = ptConsta  
    fldStoreName.MemberProperties("Store Sqft").DisplayIn = ptConsta  
  
    ' The following three lines of code set the caption for the memb  
    fldStoreName.MemberProperties("Store Manager").Caption = "Manage  
    fldStoreName.MemberProperties("Store Type").Caption = "Store Typ  
    fldStoreName.MemberProperties("Store Sqft").Caption = "Size in S  
End Sub
```



# DisplayInFieldList Property

-

Returns or sets whether the specified field set or total appears in the PivotTable Field List dialog box. Set this property to **False** to prevent the specified field set or total from appearing in the PivotTable Field List dialog box. The default value is **True**. Read/write **Boolean**.

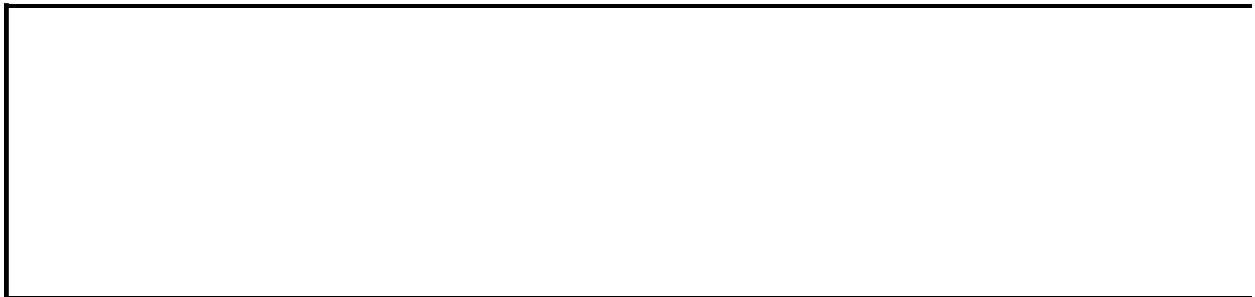
*expression*.**DisplayInFieldList**

*expression* Required. An expression that returns a **PivotFieldSet** or **PivotTotal** object.

## Example

This example adds a new total to PivotTable1. The new total is formatted to display as a percentage of the parent row field, and will not appear in the PivotTable Field List dialog box.

```
Sub Add_Total()  
  
    Dim vwView  
    Dim ptConstants  
    Dim totNewTotal  
  
    Set vwView = PivotTable1.ActiveView  
    Set ptConstants = PivotTable1.Constants  
  
    ' Add a new total named "Total Budget" to the current view.  
    Set totNewTotal = vwView.AddTotal("Total Budget", vwView.FieldSe  
        ptConstants.plFunctionSum)  
  
    ' Insert the newly created total into the detail area of the Piv  
    vwView.DataAxis.InsertTotal totNewTotal  
  
    ' Show the totals as a percentage of the parent row field.  
    totNewTotal.ShowAs = ptConstants.plShowAsPercentOfRowParent  
  
    ' Do not display the new total in the PivotTable Field List dial  
    totNewTotal.DisplayInFieldList = False  
  
End Sub
```



# DisplayOfficeLogo Property

**True** to display the Microsoft Office logo on the toolbar. Read/write **Boolean**.

*expression*.**DisplayOfficeLogo**

*expression* Required. An expression that returns a **ChartSpace**, **PivotTable**, or **Spreadsheet** object.



## Example

This example hides the Microsoft Office logo on PivotTable1's toolbar.

```
PivotTable1.DisplayOfficeLogo = False
```



# DisplayPropertyToolbox Property

-

Set this property to **True** to display the Command and Options window.  
Read/write **Boolean**.

---

*expression*.**DisplayPropertyToolbox**

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

# DisplayRowHeadings Property

-

Determines whether row headings are displayed in the specified window. Set this property to **False** to hide the row headers. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayRowHeadings**

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

## Example

This example hides the row and column headings in the active window of Spreadsheet1.

```
Sub HideHeadings()  
    Spreadsheet1.ActiveWindow.DisplayColumnHeadings = False  
    Spreadsheet1.ActiveWindow.DisplayRowHeadings = False  
End Sub
```



# DisplayScreenTips Property

-

Returns or sets whether ScreenTips are displayed. Set this property to **False** to prevent the display of ScreenTips. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayScreenTips**

*expression* Required. An expression that returns a **PivotTable** or **ChartSpace** object.

## Example

This example disables ScreenTips for PivotTable1.

```
PivotTable1.DisplayScreenTips = False
```



# DisplayTitleBar Property

**True** if the title bar on the specified spreadsheet is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayTitleBar**

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

## Example

This example hides the spreadsheet's title bar.

```
Spreadsheet1.DisplayTitleBar = False
```





# DisplayToolbar Property

**True** if the toolbar on the specified spreadsheet, chartspace, or PivotTable list is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayToolbar**

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

## Remarks

Hiding the toolbar does not change the height of the PivotTable list, but it does cause a layout change. The width is unchanged unless the [AutoFit](#) property is **True** and the PivotTable list was sized wider than the necessary to accommodate the toolbar.

## Example

This example hides the spreadsheet's toolbar.

```
Spreadsheet1.DisplayToolbar = False
```



# DisplayTotal Property

-

Returns whether or not totals are displayed for the specified **PivotResultGroupField** object. Read-only **Boolean**.

*expression*.**DisplayTotal**

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

---

---

# DisplayVerticalScrollBar Property

**True** if the vertical scroll bar on the specified spreadsheet is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayVerticalScrollBar**

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

## Example

This example hides the spreadsheet's vertical scrollbar.

```
Spreadsheet1.ActiveWindow.DisplayVerticalScrollBar = False
```



# DisplayWorkbookTabs Property

**True** if the workbook tabs are displayed. Setting this property to **False** for a single **Window** object sets this property to **False** for all windows in the workbook. The default value is **True**. Read/write **Boolean**.

*expression*.**DisplayWorkbookTabs**

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



# DisplayZeros Property

**True** if zero values are displayed. Set this property to **False** if you want to hide all zero values in the specified window. Read/write **Boolean**.

*expression*.**DisplayZeros**

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

---

---



# Divisions Property

-

Returns or sets a **Long** value that represents the number of divisions that appear in the legend for the specified segment. Read/write.

*expression*.**Divisions**

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

## Example

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created that displays the larger values in the chart with a darker shade of blue.

```
Sub Window_Onload()  
  
    Dim serSeries1  
    Dim segSegment1  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                  "Catalog=Northwind;Data Source=Da  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind Chartspace1 to the Quanti  
    ' int he Order details table.  
    ChartSpace1.SetData chConstants.chDimCategories, chConstants.chD  
    ChartSpace1.SetData chConstants.chDimValues, chConstants.chDataB  
  
    ' Create a format map.  
    ChartSpace1.SetData chConstants.chDimFormatValues, chConstants.c  
  
    ' Set a variable to the first series in the first chart in Chart  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a segment to the format map.  
    Set segSegment1 = serSeries1.FormatMap.Segments.Add  
  
    ' Measure the segment boundaries based upon a percentage.  
    segSegment1.Begin.ValueType = chConstants.chBoundaryValuePercent  
    segSegment1.End.ValueType = chConstants.chBoundaryValuePercent  
  
    ' Set the beginning value to 0%, and the ending value to 100%.  
    segSegment1.Begin.Value = 0  
    segSegment1.End.Value = 1  
  
    ' Format the interior of the matching values.  
    segSegment1.Begin.Interior.Color = "White"  
    segSegment1.End.Interior.Color = "Blue"
```

```
segSegment1.HasDiscreteDivisions = True
```

```
segSegment1.Divisions = 3
```

```
End Sub
```



▼ [Show All](#)

# DrawType Property

-  
Returns a [ChartDrawModesEnum](#) constant indicating which drawing pass is being performed. Read-only.

ChartDrawModesEnum can be one of these ChartDrawModesEnum constants.

**chDrawModeHitTest**

**chDrawModePaint**

**chDrawModeScale** Not supported for this property.

**chDrawModeSelection**

*expression*.**DrawType**

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

---

▼ [Show All](#)

# EditMode Property

-

Returns a [PivotEditModeEnum](#) constant that indicates whether the PivotTable list is currently in edit mode. Read-only.

PivotEditModeEnum can be one of these PivotEditModeEnum constants.

**plEditInProgress**

**plEditNone**

*expression*.**EditMode**

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

--

# ElementExtensions Property

-  
Returns the [ElementExtensions](#) object for the data source control.

*expression*.**ElementExtensions**

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





# ElementID Property

-  
Specifies the ID tag for the HTML element used with the specified extension.  
Read-only **String**.

*expression*.**ElementID**

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

--

# EnableAutoFilter Property

-  
**True** if the specified worksheet can be filtered. This property has no effect if the **Protection Enabled** property is set to **False**. Read/write **Boolean**.

*expression.EnableAutoFilter*

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

## Example

This example enables AutoFilter for the active worksheet, and then protects the worksheet in Spreadsheet1.

```
Sub ProtectWorksheet_EnableAutoFilter
    Spreadsheet1.ActiveSheet.EnableAutoFilter = True
    Spreadsheet1.ActiveSheet.Protection.Enabled = True
End Sub
```



# Enabled Property

-  
**OCCommand** object: Returns a **Boolean** that represents whether or not the specified command is enabled. Read only.

**Protection** object: Returns or sets a **Boolean** that determines whether or not protection is enabled for the specified worksheet. Set this property to **True** to protect the worksheet. If this property is set to **False**, the other **Protection** object property settings are ignored. Read/write.

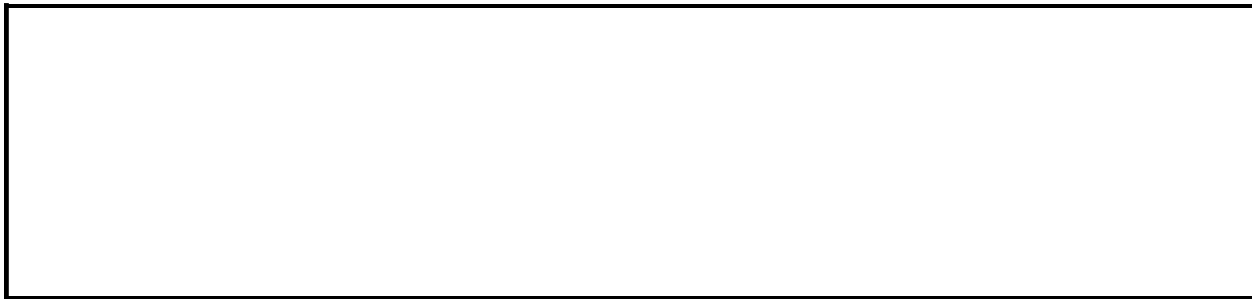
*expression*.**Enabled**

*expression* Required. An expression that returns an **OCCommand** or **Protection** object.

## Example

This example prevents a user from inserting rows on the active worksheet.

```
Sub PreventInsertingRows()  
    Dim pt  
    Set pt = Spreadsheet1.ActiveSheet.Protection  
    pt.AllowInsertingRows = False  
    pt.Enabled = True  
End Sub
```



# EnableEvents Property

**True** if events are enabled for the spreadsheet or the chart workspace.  
Read/write **Boolean**.

*expression*.**EnableEvents**

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

## Example

This example prevents spreadsheet event procedures from running.

```
Spreadsheet1.EnableEvents = False
```



# EnableResize Property

-  
**True** if the spreadsheet control can be resized by the user. Set this property to **False** to prevent the user from resizing the spreadsheet control. The default value is **True**. Read/write **Boolean**.

*expression*.**EnableResize**

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



## Example

This example sets the height and width of the spreadsheet control and then prevents the user from resizing the control.

```
Sub Size_Spreadsheet()  
    ' Set the height of the spreadsheet control.  
    Spreadsheet1.Height = 4000  
  
    ' Set the width of the spreadsheet control.  
    Spreadsheet1.Width = 6000  
  
    ' Prevent the user from resizing the spreadsheet control.  
    Spreadsheet1.ActiveWindow.EnableResize = False  
End Sub
```



# EnableUndo Property

-  
**True** if the undo functionality is enabled for the specified spreadsheet. Setting the **EnableUndo** property to **False** disables the undo functionality of the spreadsheet component. Read/write **Boolean**.

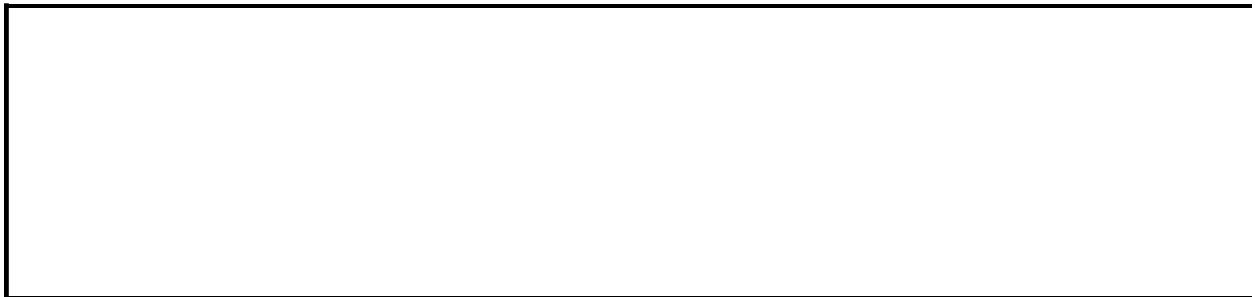
*expression*.**EnableUndo**

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

## Example

This example creates an undo block containing code that sets the number format and font for the cell D10.

```
Sub Undo_Block()  
    Dim rngCurrent  
  
    ' Enable undo.  
    Spreadsheet1.EnableUndo = True  
  
    ' Start an undo block.  
    Spreadsheet1.BeginUndo  
  
        Set rngCurrent = Spreadsheet1.Worksheets("sheet1").Range("D10")  
  
        ' The following three lines of code apply  
        ' various formatting to cell D10.  
        rngCurrent.NumberFormat = "0.###"  
        rngCurrent.Font.Color = "Blue"  
        rngCurrent.Font.Name = "Times New Roman"  
  
    ' End the undo block.  
    Spreadsheet1.EndUndo  
End Sub
```



▼ [Show All](#)

# End Property

## ► [End property as it applies to the \*\*Range\*\* object.](#)

Returns a **Range** object that represents the cell at the end of the region that contains the specified source range. Using this property is equivalent to pressing the CTRL+UP ARROW, CTRL+DOWN ARROW, CTRL+LEFT ARROW, or CTRL+RIGHT ARROW key combinations.

*expression*.**End**(*Direction*)

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

*Direction* Required [XlDirection](#). The direction in which to move.

XlDirection can be one of these XlDirection constants.

**xlDown**

**xlToLeft**

**xlToRight**

**xlUp**

## ► [End property as it applies to the \*\*ChSegment\*\* object.](#)

Returns a **ChSegmentBoundary** object that represents the end of a segment boundary on a format map.

*expression*.**End**

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

## Remarks

Use the **Value** property of the returned **ChSegmentBoundary** object to set the ending value for the specified segment of the format map. Use the **Interior**, **Line**, and **Border** properties to format the segment boundary.

## Example

### ► [As it applies to the \*\*Range\*\* object.](#)

This example selects a range of cells starting at the active cell, moving down in direction until a blank cell is reached.

```
Sub SelectContiguousCells()  
  
    Dim ssConstants  
    Dim rngStartCell  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    Set rngStartCell = Spreadsheet1.ActiveCell  
  
    Spreadsheet1.ActiveSheet.Range(rngStartCell, _  
        rngStartCell.End(ssConstants.xlDown)).Select  
  
End Sub
```

### ► [As it applies to the \*\*ChSegment\*\* object.](#)

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then a format map is created that highlights the bottom 10% of the values in red and the top 20% of values in green.

```
Sub Window_Onload()  
  
    Dim serseries1  
    Dim segBottom10Pct  
    Dim segTop20Pct  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Secur  
        "User ID=sa;Initial Catalog=Northwind;Data Sour  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind Chartspace1 to the Quanti  
    ' in the Order details table.  
    ChartSpace1.SetData chConstants.chDimCategories, chConstants.chD
```

```

ChartSpace1.SetData chConstants.chDimValues, chConstants.chDataB

' Create a format map.
ChartSpace1.SetData chConstants.chDimFormatValues, chConstants.c

' Set a variable to the first series in the first chart in Chart
Set serseries1 = ChartSpace1.Charts(0).SeriesCollection(0)

' Add a segment to the format map. This segment will
' represent the bottom 10% of values in the chart.
Set segBottom10Pct = serseries1.FormatMap.Segments.Add

' Measure the segment boundaries based upon a percentage.
segBottom10Pct.Begin.ValueType = chConstants.chBoundaryValuePerc
segBottom10Pct.End.ValueType = chConstants.chBoundaryValuePercen

' Set the beginning value to 0%, and the ending value to 10%.
segBottom10Pct.Begin.Value = 0
segBottom10Pct.End.Value = 0.1

' Format the interior of the matching values.
segBottom10Pct.Begin.Interior.Color = "red"
segBottom10Pct.End.Interior.Color = "red"

' Add a segment to the format map. This segment will
' represent the top 20% of values in the chart.
Set segTop20Pct = serseries1.FormatMap.Segments.Add

' Measure the segment boundaries based upon a percentage.
segTop20Pct.Begin.ValueType = chConstants.chBoundaryValuePercent
segTop20Pct.End.ValueType = chConstants.chBoundaryValuePercent

' Set the beginning value to 80%, and the ending value to 100%.
segTop20Pct.Begin.Value = 0.8
segTop20Pct.End.Value = 1

' Format the interior of the matching values.
segTop20Pct.Begin.Interior.Color = "green"
segTop20Pct.End.Interior.Color = "green"

```

End Sub





# EndStyle Property

-  
Returns or sets the end style for error bars. Read/write [ChartEndStyleEnum](#).

ChartEndStyleEnum can be one of these ChartEndStyleEnum constants.

**chEndStyleCap** *default*

**chEndStyleNone**

*expression*.EndStyle

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

## Example

This example adds error bars to all of the series in the first chart in ChartSpace1 and then sets the error amount and end style.

```
Sub Add_Error_Bars()  
    Dim ebErrorBars  
    Dim serChartSeries  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Loop through all of the series in the first chart  
    ' in ChartSpace1.  
    For Each serChartSeries in ChartSpace1.Charts(0).SeriesCollection  
        ' Add error bars to the current series.  
        Set ebErrorBars = serChartSeries.ErrorBarsCollection.Add  
  
        ' Set the error bars to be a percentage of the value.  
        ebErrorBars.Type = chConstants.chErrorBarTypePercent  
  
        ' Set the percentage amount.  
        ebErrorBars.Amount = 0.05  
  
        ' Set the end style of the error bars.  
        ebErrorBars.EndStyle = chConstants.chEndStyleNone  
    Next  
End Sub
```



# EntireColumn Property

-  
Returns a [Range](#) object that represents the entire column (or columns) containing the specified range. Read-only.

*expression*.**EntireColumn**

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

## Example

This example sets the font color for the entire column containing the active cell.

```
Spreadsheet1.ActiveCell.EntireColumn.Font.Color = "green"
```



# EntireRow Property

-

Returns a [Range](#) object that represents the entire row (or rows) containing the specified range. Read-only.

*expression*.**EntireRow**

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

## Example

This example sets the font color for the entire row containing the active cell.

```
Spreadsheet1.ActiveCell.EntireRow.Font.Color = "red"
```



# Error Property

-

Returns an ADO **Error** object that contains error information about the specified event. Read-only.

---

*expression*.**Error**

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



# ErrorBarsCollection Property

-

Returns a **ChErrorBarsCollection** collection that contains a **ChErrorBars** object for each set of error bars in the specified series. A series can have only one set of error bars, so the **ChErrorBarsCollection** collection never contains more than one object. Read-only.

*expression*.**ErrorBarsCollection**

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

## Example

This example adds error bars to all of the series in the first chart in ChartSpace1 and then sets the error amount and end style.

```
Sub Add_Error_Bars()  
    Dim ebErrorBars  
    Dim serChartSeries  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Loop through all of the series in the first chart  
    ' in ChartSpace1.  
    For Each serChartSeries in ChartSpace1.Charts(0).SeriesCollection  
        ' Add error bars to the current series.  
        Set ebErrorBars = serChartSeries.ErrorBarsCollection.Add  
  
        ' Set the error bars to be a percentage of the value.  
        ebErrorBars.Type = chConstants.chErrorBarTypePercent  
  
        ' Set the percentage amount.  
        ebErrorBars.Amount = 0.05  
  
        ' Set the end style of the error bars.  
        ebErrorBars.EndStyle = chConstants.chEndStyleNone  
    Next  
End Sub
```



# ExcludedMembers Property

-

Returns or sets the members that you do not want to be displayed in the specified field. This property can be set to a single member or a **Variant** array of members. The members can be passed as one or more **PivotMember** objects, member names, or unique member names.

*expression*.**ExcludedMembers**

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

## Remarks

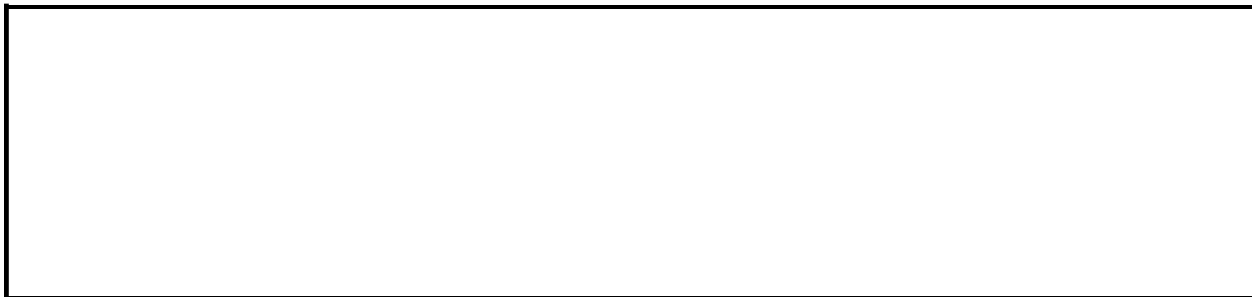
Members that are explicitly excluded still appear in the PivotTable list if one or more of their children are included. Setting this property clears all previous settings of this property for the specified field. You can set the

**ExcludedMembers** property to **Empty** (`ExcludedMembers = Empty`) or to a zero-length **Variant** array (`ExcludedMembers = Array()`) to clear the included members list for the specified field.

## Example

This example sets the included and excluded members of the Store State and Store City fields in PivotTable1.

```
Sub Member_Filtering()  
  
    Dim fldStoreCity  
    Dim fldStoreState  
    Dim ptView  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Store State field.  
    Set fldStoreState = ptView.FieldSets("Store").Fields("Store State")  
  
    ' Set a variable to the Store City field.  
    Set fldStoreCity = ptView.FieldSets("Store").Fields("Store City")  
  
    ' Exclude California and Washington from the Store State field.  
    fldStoreState.ExcludedMembers = Array("CA", "WA")  
  
    ' Include members of the Store City field. Note that the cities  
    ' in states that have been excluded by the previous line. Since  
    ' Store State is a parent to Store City, then the excluded state  
    ' are displayed in the PivotTable.  
    fldStoreCity.IncludedMembers = Array("Los Angeles", "San Diego",  
                                          "Seattle", "Spokane")  
  
End Sub
```



▼ [Show All](#)

# ExpandDetails Property

-

Returns or sets a [PivotTableExpandEnum](#) constant that represents whether or not fields are expanded when added to the detail area of a PivotTable list.

Read/write.

PivotTableExpandEnum can be one of these PivotTableExpandEnum constants.

**plExpandAlways** Fields are always expanded when added to the detail area of a PivotTable list. This setting does not prevent the user from collapsing the field once it has been added to the PivotTable list.

**plExpandAutomatic** *default* For relational data sources, fields and their members are expanded. For multidimensional data sources, fields and their members are not expanded.

**plExpandNever** Fields are never added to the detail area of a PivotTable list. This setting does not prevent the user from expanding the field once it has been added to the PivotTable list.

*expression*.**ExpandDetails**

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

## Example

This example sets PivotTable1 so that fields are never expanded when they are added to the PivotTable list.

```
Sub NeverExpand()  
    Dim pvtView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view.  
    Set pvtView = PivotTable1.ActiveView  
  
    ' Always expand fields when they are added  
    ' to a PivotTable list.  
    pvtView.ExpandMembers = ptConstants.plExpandNever  
  
    ' Always expand fields when they are added  
    ' to the detail area PivotTable list.  
    pvtView.ExpandDetails = ptConstants.plExpandNever  
End Sub
```





# Expanded Property

-  
**PivotCell** object: **True** if the detail grid is being displayed for the specified cell. If this property is set to **False**, the detail grid is not displayed but aggregates are displayed if they are available. Read/write **Boolean**.

**PivotAxisMember** object: **True** if child members or details are being displayed for the specified member; **False** if child members or details are collapsed. Read/write **Boolean**.

**PivotField** object: Returns or sets the expanded state of all the members of the specified field at the same time. Setting the **Expanded** property for the field immediately sets the **Expanded** property for all the members associated with the field. Setting the **Expanded** property for a single member of the field does not affect the **Expanded** property for the field itself. Read/write **Boolean**.

*expression*.**Expanded**

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

--

# ExpandedByDefault Property

**True** if the specified group level is expanded by default. The default value is **False**. Read/write **Boolean**.

*expression*.**ExpandedByDefault**

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

--

▼ [Show All](#)

# ExpandMembers Property

Returns or sets a [PivotTableExpandEnum](#) constant that represents whether or not fields and their members are expanded when a field is added to a PivotTable list. Read/write.

PivotTableExpandEnum can be one of these PivotTableExpandEnum constants.

**plExpandAlways** Fields and their members are always expanded when added to a PivotTable list. This setting does not prevent the user from collapsing the field once it has been added to the PivotTable list.

**plExpandAutomatic** *default* For relational data sources, fields and their members are expanded. For multidimensional data sources, fields and their members are not expanded.

**plExpandNever** Fields and their members are never expanded when added to a PivotTable list. This setting does not prevent the user from expanding the field once it has been added to the PivotTable list.

*expression*.**ExpandMembers**

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

## Example

This example sets PivotTable1 so that fields are never expanded when they are added to the PivotTable list.

```
Sub NeverExpand()  
    Dim pvtView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view.  
    Set pvtView = PivotTable1.ActiveView  
  
    ' Always expand fields when they are added  
    ' to a PivotTable list.  
    pvtView.ExpandMembers = ptConstants.plExpandNever  
  
    ' Always expand fields when they are added  
    ' to the detail area PivotTable list.  
    pvtView.ExpandDetails = ptConstants.plExpandNever  
End Sub
```



▼ [Show All](#)

# Explosion Property

## ► [Explosion property as it applies to the \*\*ChPoint\*\* object.](#)

Returns or sets the explosion value for the specified pie-chart or doughnut-chart slice. The valid range is from 0 through 1000. The explosion value is equal to a percentage of the chart's radius. Note that setting this property to 100 moves the tips of the slices away from the center of the chart to a distance equal to the chart's radius. Use this property to highlight a particular pie slice. Read/write **Long**.

*expression*.**Explosion**

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

## ► [Explosion property as it applies to the \*\*ChSeries\*\* object.](#)

Returns or sets the explosion value for the specified pie-chart or doughnut-chart series. The valid range is from 0 through 1000. The explosion value is equal to a percentage of the chart's radius. Note that setting this property to 100 moves the tips of the slices away from the center of the chart to a distance equal to the chart's radius. Read/write **Long**.

*expression*.**Explosion**

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

## Example

► [As it applies to the \*\*ChPoint\*\* object.](#)

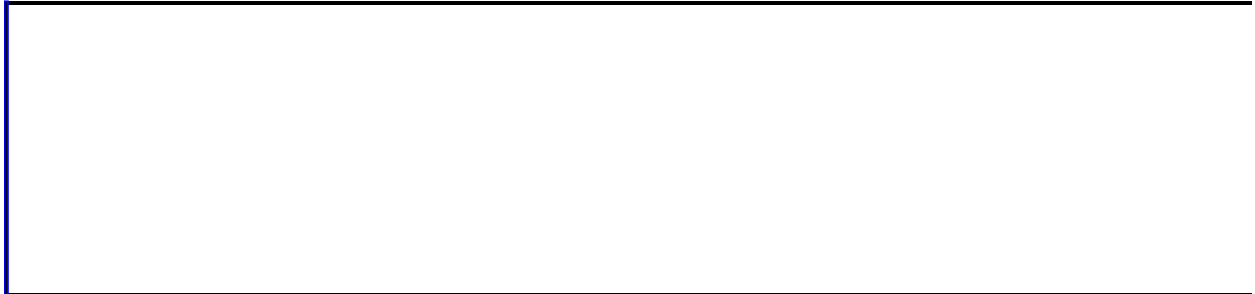
This example moves the tips of the third point in the specified series to a distance equal to one-tenth the radius of the chart.

```
ChartSpace1.Charts(0).SeriesCollection(0).Points(2).Explosion = 10
```

► [As it applies to the \*\*ChSeries\*\* object.](#)

This example moves the tips of the pie-chart or doughnut-chart slices in the specified series to a distance equal to one-tenth the radius of the chart.

```
ChartSpace1.Charts(0).SeriesCollection(0).Explosion = 10
```





# Expression Property

-

Returns or sets a **String** that represents the expression used to calculate the specified calculated field or calculated total. The expression must be compatible with the Jet expression service. Read/write.

*expression*.**Expression**

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

## Remarks

The **Expression** property will return a blank string if it is not used with a calculated field or calculated total.

## Example

The following example displays the current expression used for a calculated field named "Variance" in PivotTable1. When you edit the expression and then click OK, the new expression is assigned to the Variance field.

```
Sub Change_Expression()  
  
    Dim vwView  
    Dim cfCalcField  
    Dim strCurrentExpression  
    Dim strNewExpression  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' Set a variable to the calculated field.  
    Set cfCalcField = _  
        vwView.Fieldsets("Variance").Fields("Variance")  
  
    ' Set a variable to the current expression used in the  
    ' Variance field.  
    strCurrentExpression = cfCalcField.Expression  
  
    ' Display an input box that contains the current expression for  
    ' Variance field. Edit the expression and then click OK.  
    strNewExpression = InputBox("Edit the expression used for the ca  
        "field and then click OK.", , strCurrentExpression)  
  
    ' Set the expression used to calculate the Variance field to the  
    ' expression entered in the input box.  
    cfCalcField.Expression = strNewExpression  
  
End Sub
```

---

# ExtrudeAngle Property

-

Returns or sets a **Double** specifying the direction of extrusion for a three-dimensional chart. This property is valid only if the **ProjectionMode** property of the chart has been set to **chProjectionModeOrthographic**. Valid values range from 0 to 360. The default value is 45. Read/write.

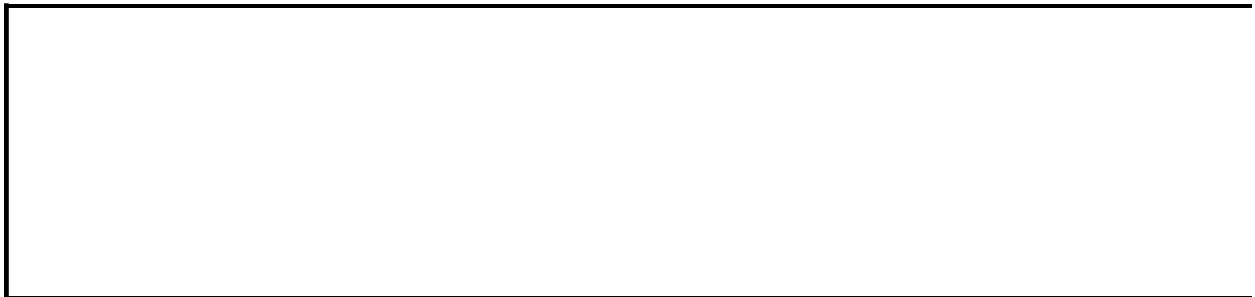
*expression*.**ExtrudeAngle**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then sets the projection mode of the chart.

```
Sub SetExtrudeAngle()  
  
    Dim cht3DColumn  
    Dim chConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Column chart.  
    cht3DColumn.Type = chChartTypeColumnClustered3D  
  
    ' Set the projection mode to orthographic.  
    cht3DColumn.ProjectionMode = chConstants.chProjectionModeOrthogr  
  
    ' Set the extrusion angle.  
    cht3DColumn.ExtrudeAngle = 75  
  
End Sub
```



# Field Property

-  
Returns a [PivotField](#) object that represents the field associated with the specified member or total.

*expression*.**Field**

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

--

# FieldLabelBackColor Property

-

Returns or sets the back color used for field labels for rows, columns, and filters. The default value is 25% gray. Read/write **Variant**.

*expression*.**FieldLabelBackColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---



# FieldLabelFont Property

-

Returns a [PivotFont](#) object that represents the field label font for rows, columns, and filters. Read-only.

---

*expression*.**FieldLabelFont**

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

# FieldLabelForeColor Property

-

Returns or sets the foreground color used for field labels for rows, columns, and filters. Read/write **Variant**.

*expression*.**FieldLabelForeColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

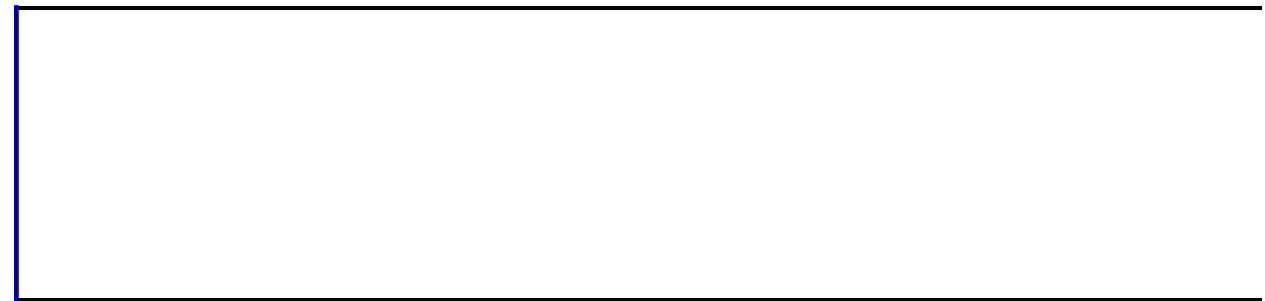
This property always returns the color as a **Long** value representing a red-green-blue color value.

---

# FieldLabelHeight Property

-

Returns the height of the field labels for rows, columns, and filters. Read-only **Long**.



*expression*.**FieldLabelHeight**

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

# Fields Property

-  
Returns the [PivotFields](#) collection for the specified field set.

*expression*.**Fields**

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

--

# FieldSet Property

-

Returns a [PivotFieldSet](#) object that represents the field set to which the specified field belongs.

*expression*.**FieldSet**

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

--

# FieldSets Property

-  
Returns a [PivotFieldSets](#) object that contains the field sets associated with the specified axis or view.

*expression*.**FieldSets**

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

--

▼ [Show All](#)



# FieldType Property

-  
Returns the field type for the specified field. Read-only [DscFieldTypeEnum](#).

DscFieldTypeEnum can be one of these DscFieldTypeEnum constants.

**dscCalculated**

**dscGrouping**

**dscOutput**

**dscParameter**

*expression*.**FieldType**

*expression* Required. An expression that returns a **PageField** property.

---

---

▼ [Show All](#)

# FillType Property

Returns a [ChartFillTypeEnum](#) constant indicating the type of fill used for the specified **ChInterior** object. Read-only.

ChartFillTypeEnum can be one of these ChartFillTypeEnum constants.

**chFillGradientOneColor**

**chFillGradientPresetColors**

**chFillGradientTwoColors**

**chFillPatterned**

**chFillSolid**

**chFillTexturePreset**

**chFillTextureUserDefined**

*expression*.**FillType**

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

## Remarks

You can use the following methods to set the type of fill for a **ChInterior** object: **SetOneColorGradient**, **SetPatterned**, **SetPresetGradient**, **SetSolid**, **SetTextured**, and **SetTwoColorGradient**.

# Filter Property

-

Returns the ADO filter string for the recordset that corresponds to the data access page. Read/write **Variant**.

*expression*.**Filter**

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

--

# FilterAxis Property

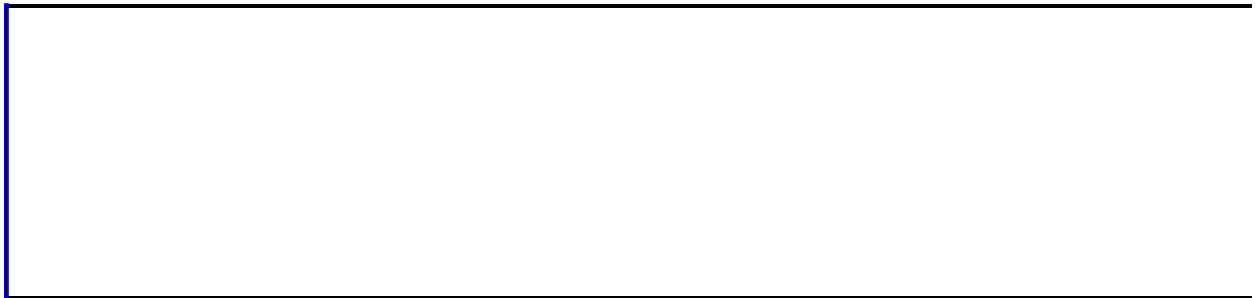
-

**PivotData** object: Returns a **PivotResultFilterAxis** object that represents the filter axis.

**PivotView** object: Returns a **PivotFilterAxis** object that represents the filter axis.

*expression*.**FilterAxis**

*expression* Required. An expression that returns a **PivotData** or **PivotView** object.



# FilterCaption Property

-

Returns a **String** that represents the caption displayed just below a field set on the filter axis. Read-only.

*expression*.**FilterCaption**

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

## Remarks

If the user has selected only one item in the specified field, this property will return that item. If the user has selected multiple items in the field, this property will most likely return "(Multiple Items)".



# FilterContext Property

-

Returns a **PivotField** object that represents the context by which the conditional filter will be evaluated.

*expression*.**FilterContext**

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

## Remarks

When you apply a conditional filter to a field, the filter can apply to that field, or one of its parent fields in the field set. For example, assume that your PivotTable list contains a field set named Store. The Store field set contains the following fields: Country, Region, City, and Store Number. The Country field is a parent to the Region field, the Region field is a parent to the City field, and the City field is a parent to the Store Number field.

Now you want to find the top five most profitable cities. If you set this property to the City field, then the top five most profitable cities will be displayed in the PivotTable list. If you set this property to the Region field, then the top five cities will be returned for each region.

Setting this property to a child of the field that you are applying a conditional filter to will result in a run-time error. For example, you cannot set this property to the Store Number field when you are applying a conditional filter to the City field.

## Example

This example displays the least profitable city in each state.

```
Sub LeastProfitableByState()  
  
    Dim ptView  
    Dim ptConstants  
    Dim fldFilterField  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view of the PivotTable list.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the field that is to be filtered.  
    Set fldFilterField = PivotTable1.ActiveData.RowAxis.Fields("Store City")  
  
    ' Filter the stores based on profit.  
    Set fldFilterField.FilterOn = ptView.Totals("Profit")  
  
    ' Set the function used to filter the stores.  
    fldFilterField.FilterFunction = ptConstants.plFilterFunctionBottom  
  
    ' Display the least profitable store.  
    fldFilterField.FilterFunctionValue = 1  
  
    ' Set the context of the filter. Although we are filtering based  
    ' the Store City field, setting the filter context to the Store  
    ' field means that the least profitable store from each state will  
    ' be displayed.  
    Set ptView.FieldSets("Store").Fields("Store City").FilterContext =  
        ptView.FieldSets("Store").Fields("Store State")  
  
End Sub
```



# FilterCrossJoins Property

-

Returns or sets a **Boolean** that determines how the PivotTable control processes empty members when retrieving the data for the current view. The default value is **True**. Read/write.

*expression*.**FilterCrossJoins**

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

## Remarks

In most cases, do not want to set this property to **False**. However, if your OLAP cube contains a field set where the top member is empty, then you may want to set this property to **False**.

▼ [Show All](#)

# FilterFunction Property

► [FilterFunction property as it applies to the \*\*PivotField\*\* object.](#)

Returns or sets the filter function for the specified field. Read/write [PivotFieldFilterFunctionEnum](#).

PivotFieldFilterFunctionEnum can be one of these PivotFieldFilterFunctionEnum constants.

**plFilterFunctionBottomCount**

**plFilterFunctionBottomPercent**

**plFilterFunctionBottomSum**

**plFilterFunctionExpression**

**plFilterFunctionMemberSet**

**plFilterFunctionNone**

**plFilterFunctionTopCount**

**plFilterFunctionTopPercent**

**plFilterFunctionTopSum**

*expression*.**FilterFunction**

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

► [FilterFunction property as it applies to the \*\*Criteria\*\* object.](#)

Returns or sets how the specified criteria work with the AutoFilter. If this property is set to **ssFilterFunctionInclude**, the specified items are displayed in the filtered list and all other items are removed. If this property is set to **ssFilterFunctionExclude**, the specified items are removed from the filtered list and all other items are displayed. Read/write [SheetFilterFunction](#).

SheetFilterFunction can be one of these SheetFilterFunction constants.

**ssFilterFunctionExclude**

## **ssFilterFunctionInclude**

*expression*.**FilterFunction**

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

--



# FilterFunctionValue Property

Returns or sets a **Variant** representing the value used to filter a field. The type of value will vary based on the current setting of the **FilterFunction** property. Use the following table to determine an appropriate value for this property. Read/write.

FilterFunction setting	Appropriate value range
<b>plFilterFunctionBottomCount</b>	<b>Integer</b> value representing how many of the bottom members that you want to display.
<b>plFilterFunctionBottomPercent</b>	<b>Double</b> value between 0 and 1 representing the percentage of members that you want to display.
<b>plFilterFunctionBottomSum</b>	
<b>plFilterFunctionNone</b>	
<b>plFilterFunctionTopCount</b>	<b>Integer</b> value representing how many of the top members that you want to display.
<b>plFilterFunctionTopPercent</b>	<b>Double</b> value between 0 and 1 representing the percentage of members that you want to display.
<b>plFilterFunctionTopSum</b>	

*expression*.**FilterFunctionValue**

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

## Example

This example applies a conditional filter to the Store City field based on the Profit total. The three most profitable stores are displayed.

```
Sub TopThreeStores()  
  
    Dim ptView  
    Dim ptConstants  
    Dim fldFilterField  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view of the PivotTable list.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the field that is to be filtered.  
    Set fldFilterField = PivotTable1.ActiveData.RowAxis.Fields("Store City")  
  
    ' Filter the stores based on profit.  
    Set fldFilterField.FilterOn = ptView.Totals("Profit")  
  
    ' Set the function used to filter the stores.  
    fldFilterField.FilterFunction = ptConstants.plFilterFunctionTopC  
  
    ' Display the three most profitable stores.  
    fldFilterField.FilterFunctionValue = 3  
  
End Sub
```



# FilterMode Property

**True** if any worksheet rows are currently hidden by AutoFilter criteria. The default value is **False**. Read-only **Boolean**.

*expression*.**FilterMode**

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

## Remarks

If the AutoFilter drop-down arrows are visible but no rows are currently filtered (all rows are visible), the [AutoFilterMode](#) property is **True** and the **FilterMode** property is **False**.

## Example

This example turns on the AutoFilter for the range A1:C20, sets filters for columns A and C, and then applies the filters. The **FilterMode** property returns **False** until the AutoFilter criteria has been applied to the list.

```
Sub Apply_AutoFilter()  
    Dim afFilters  
    Dim afCol1  
    Dim afCol3  
  
    ' Turn on AutoFilter.  
    Spreadsheet1.Worksheets("Sheet1").Range("A1:C20").AutoFilter  
  
    ' Set a variable to the AutoFilter object.  
    Set afFilters = Spreadsheet1.Worksheets("sheet1").AutoFilter  
  
    Set afCol1 = afFilters.Filters(1)  
    Set afCol3 = afFilters.Filters(3)  
  
    ' Add a criteria that excludes blue from column A.  
    afCol1.Criteria.Add "blue"  
  
    ' Add a criteria that excludes green from column A.  
    afCol1.Criteria.Add "green"  
  
    ' Add a criteria that excludes yellow from column c.  
    afCol3.Criteria.Add "yellow"  
  
    ' At this point, the FilterMode property is False  
    ' because the AutoFilter criteria has not been applied.  
    MsgBox Spreadsheet1.Worksheets("Sheet1").FilterMode  
  
    ' Apply the criteria.  
    afFilters.Apply  
  
    ' The FilterMode property is now True since you  
    ' have hidden several rows in the list.  
    MsgBox Spreadsheet1.Worksheets("Sheet1").FilterMode  
End Sub
```



# FilterOn Property

-

Returns a **PivotTotal** object that represents the total to use when conditionally filtering a field.

*expression*.**FilterOn**

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

## Example

This example applies a conditional filter to the Store City field based on the Profit total. The three most profitable stores are displayed.

```
Sub TopThreeStores()  
  
    Dim ptView  
    Dim ptConstants  
    Dim fldFilterField  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view of the PivotTable list.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the field that is to be filtered.  
    Set fldFilterField = PivotTable1.ActiveData.RowAxis.Fields("Store City")  
  
    ' Filter the stores based on profit.  
    Set fldFilterField.FilterOn = ptView.Totals("Profit")  
  
    ' Set the function used to filter the stores.  
    fldFilterField.FilterFunction = ptConstants.plFilterFunctionTopC  
  
    ' Display the three most profitable stores.  
    fldFilterField.FilterFunctionValue = 3  
  
End Sub
```





# FilterOnScope Property

-

This property establishes the scope that is used when conditionally filtering a field. You can pass a **String** containing the unique name of a member or an array of unique names of members. Returns a **Variant** array containing **PivotMember** objects. Read/write.

*expression*.**FilterOnScope**

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

## Remarks

The scope is based on one or more members of a different field than the field that is being filtered. For example, you may want to apply a filter to the Customer field, which has been added to the row axis of your PivotTable list, to display the three customers to whom you have made the most sales. To do this, you set the **FilterFunction** property to **plFilterFunctionTopCount**, the **FilterFunctionValue** property to 3, and the **FilterOn** property to the Total Sales total. The PivotTable list displays your three best customers.

If you need to narrow your query to a more specific set of customers, then you would use the **FilterOnScope** property. If you want to see the your top customers in the Southeast region, then you would set the **FilterOnScope** property to an expression that evaluates to the Southeast member in the Region field.

## Example

This example displays the two top-selling products in Canada.

```
Sub DisplayTopTwoCanadianSellers()  
  
    Dim objPivotView  
    Dim objPivotData  
    Dim fldProductName  
  
    ' Set a variable to the active view of the PivotTable.  
    Set objPivotView = PivotTable1.ActiveView  
  
    ' Set a variable to the active data of the PivotTable.  
    Set objPivotData = PivotTable1.ActiveData  
  
    ' Set a variable to the Product Name field, which has been  
    ' added to the row axis of the PivotTable list.  
    Set fldProductName = objPivotData.RowAxis.Fields("Product Name")  
  
    ' The following two lines of code set the PivotTable list to fil  
    ' top 2 items.  
    fldProductName.FilterFunction = PivotTable1.Constants.plFilterFu  
    fldProductName.FilterFunctionValue = 2  
  
    ' Filter based on the Unit Sales total.  
    Set fldProductName.FilterOn = objPivotView.Totals("Unit Sales")  
  
    ' Set the filter scope to include only sales in Canada.  
    fldProductName.FilterOnScope = objPivotView.FieldSets("Store").M  
  
End Sub
```



# Filters Property

-

Returns the [Filters](#) collection for the specified AutoFilter. The **Filters** collection contains one [Filter](#) object for each column in the filtered range. Read-only.

*expression*.**Filters**

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

## Example

This example turns on the AutoFilter for the range A1:C20, sets filters for columns A and C, and then applies the filters.

```
Sub Apply_AutoFilter()  
    Dim afFilters  
    Dim afCol1  
    Dim afCol3  
  
    ' Turn on AutoFilter.  
    Spreadsheet1.Worksheets("Sheet1").Range("A1:C20").AutoFilter  
  
    ' Set a variable to the AutoFilter object.  
    Set afFilters = Spreadsheet1.Worksheets("sheet1").AutoFilter  
  
    Set afCol1 = afFilters.Filters(1)  
    Set afCol3 = afFilters.Filters(3)  
  
    ' Add a criteria that excludes blue from column A.  
    afCol1.Criteria.Add "blue"  
  
    ' Add a criteria that excludes green from column A.  
    afCol1.Criteria.Add "green"  
  
    ' Add a criteria that excludes yellow from column c.  
    afCol3.Criteria.Add "yellow"  
  
    ' Apply the criteria.  
    afFilters.Apply  
End Sub
```



▼ [Show All](#)

# FindAxisMember Property

-

Finds an axis member, given a reference to the member. Returns a **PivotAxisMember** object.

*expression*.**FindAxisMember**(*Path*, *Format*)

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

***Path*** Required **String**. A variable or string that contains a reference to the member to find.

***Format*** Required [PivotMemberFindFormatEnum](#). Indicates the format used for the ***Path*** argument.

PivotMemberFindFormatEnum can be one of these  
PivotMemberFindFormatEnum constants.

**plFindFormatMember** Not supported for this property.

**plFindFormatPathHex** Not supported for this property.

**plFindFormatPathInt** Member reference is a path of indexes. For example, "1\0\5\1".

**plFindFormatPathName** Member reference is a unique name, or can be a name if unambiguous. For example, "[USA].[Oregon].[Portland]".

## Remarks

If the requested member is not found, a **PivotAxisMember** object with the **IsValid** property set to **False** is returned. This allows you to refer to a member that might later be added to the schema.

This property can be used to find a member from the top member of the specified axis. Use the **Member** property to return the top member of an axis.



▼ [Show All](#)

# FindColumnMember Property

-

Finds a column member, given a reference to the member. Returns a **PivotColumnMember** object.

*expression*.**FindColumnMember**(*Path*, *Format*)

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

***Path*** Required **String**. A variable or string that contains a reference to the member to find.

***Format*** Required [PivotMemberFindFormatEnum](#). Indicates the format used for the ***Path*** argument.

PivotMemberFindFormatEnum can be one of these PivotMemberFindFormatEnum constants.

**plFindFormatMember** Not supported for this property.

**plFindFormatPathHex** Not supported for this property.

**plFindFormatPathInt** Member reference is a path of indexes. For example, "1\0\5\1".

**plFindFormatPathName** Member reference is a unique name, or can be a name if unambiguous. For example, "[USA].[Oregon].[Portland]".

## Remarks

If the requested member is not found, a **PivotColumnMember** object with the **IsValid** property set to **False** is returned. This allows you to refer to a member that might later be added to the schema.

This property can be used to find a member from the top member of the column axis. Use the **Member** property to return the top member of the column axis.

▼ [Show All](#)

# FindMember Property

► [FindMember property as it applies to the \*\*PivotFieldSet\*\* object.](#)

Finds a member, given a reference to the member. Returns a **PivotMember** object.

*expression*.**FindMember**(*NameOrPath*, *Format*)

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

**NameOrPath** Required **Variant**. A variable or string that contains a reference to the member to find.

**Format** Optional [PivotMemberFindFormatEnum](#). Indicates the format used for the **Path** argument.

PivotMemberFindFormatEnum can be one of these  
PivotMemberFindFormatEnum constants.

**plFindFormatMember** Not supported for this property.

**plFindFormatPathHex** Not supported for this property.

**plFindFormatPathInt** Not supported for this property.

**plFindFormatPathName** Member reference is a unique name or can be a name if unambiguous. For example, "[USA].[Oregon].[Portland]".

► [FindMember property as it applies to the \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotMember\*\*, \*\*PivotPageMember\*\*, and \*\*PivotRowMember\*\* objects.](#)

Finds a member, given a reference to the member. Returns a **PivotMember** object.

*expression*.**FindMember**(*Path*, *Format*)

*expression* Required. An expression that returns one of the above objects.

***Path*** Required **String**. A variable or string that contains a reference to the member to find.

***Format*** Required [\*\*PivotMemberFindFormatEnum\*\*](#). Indicates the format used for the ***Path*** argument.

PivotMemberFindFormatEnum can be one of these  
PivotMemberFindFormatEnum constants.

**plFindFormatMember** Not supported for this property.

**plFindFormatPathHex** Not supported for this property.

**plFindFormatPathInt** Member reference is a path of indexes. For example, "1\0\5\1".

**plFindFormatPathName** Member reference is a unique name or can be a name if unambiguous. For example, "[USA].[Oregon].[Portland]".

## Remarks

If the requested member is not found, a **PivotMember** object with the **IsValid** property set to **False** is returned. This allows you to refer to a member that might later be added to the schema.

## Example

This example attempts to find a specific warehouse in the Warehouse field set. The user is alerted if the specified warehouse is not found.

```
Sub FindWarehouse()  
  
    Dim ptView  
    Dim ptConstants  
    Dim fsWarehouse  
    Dim pmFound  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Warehouse field set.  
    Set fsWarehouse = ptView.FieldSets("Warehouse")  
  
    ' Set a variable to the results of the FindMember property.  
    Set pmFound = fsWarehouse.FindMember("Quality Distribution, Inc.")  
  
    ' Check to see if the member was found.  
    If pmFound.IsValid = False Then  
  
        ' Alert the user if the member was not found.  
        MsgBox "The specified member does not exist."  
  
    End If  
  
End Sub
```





▼ [Show All](#)

# FindPageMember Property

-

Finds a page member, given a reference to the member. Returns a **PivotPageMember** object.

*expression*.**FindPageMember**(*Path*, *Format*)

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

**Path** Required **String**. A variable or string that contains a reference to the member to the member to find.

**Format** Required [PivotMemberFindFormatEnum](#). Indicates the format used for the **Path** argument.

PivotMemberFindFormatEnum can be one of these  
PivotMemberFindFormatEnum constants.

**plFindFormatMember** Not supported for this property.

**plFindFormatPathHex** Not supported for this property.

**plFindFormatPathInt** Member reference is a path of indexes. For example, "1\0\5\1".

**plFindFormatPathName** Member reference is a unique name or can be a name if unambiguous. For example, "[USA].[Oregon].[Portland]".

## Remarks

If the requested member is not found, a **PivotPageMember** object with the **IsValid** property set to **False** is returned. This allows you to refer to a member that might later be added to the schema.

This property can be used to find a member from the top member of the page axis. Use the **Member** property to return the top member of the page axis.

▼ [Show All](#)

# FindRowMember Property

-

Finds a row member, given a reference to the member. Returns a **PivotRowMember** object.

*expression*.FindRowMember(***Path***, ***Format***)

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

***Path*** Required **String**. A variable or string that contains a reference to the member to find.

***Format*** Required [PivotMemberFindFormatEnum](#). Indicates the format used for the ***Path*** argument.

PivotMemberFindFormatEnum can be one of these  
PivotMemberFindFormatEnum constants.

**plFindFormatMember** Member reference is a named path. For example, "USA\Oregon\Portland".

**plFindFormatPathHex** Member reference is a path of hexadecimal indexes.

**plFindFormatPathInt** Member reference is a path of indexes. For example, "1\0\5\1".

**plFindFormatPathName** Member reference is a unique name or can be a name if unambiguous. For example, "[USA].[Oregon].[Portland]".

## Remarks

If the requested member is not found, a **PivotRowMember** object with the **IsValid** property set to **False** is returned. This allows you to refer to a member that might later be added to the schema.

# FirstSection Property

-

Returns a [Section](#) object that represents the first section on the specified data page. Read-only.

---

*expression*.**FirstSection**

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

# FirstSliceAngle Property

-

Returns or sets the angle of the first pie-chart or doughnut-chart slice, in degrees (clockwise from vertical). Applies only to pie and doughnut charts. Read/write **Long**.

*expression*.**FirstSliceAngle**

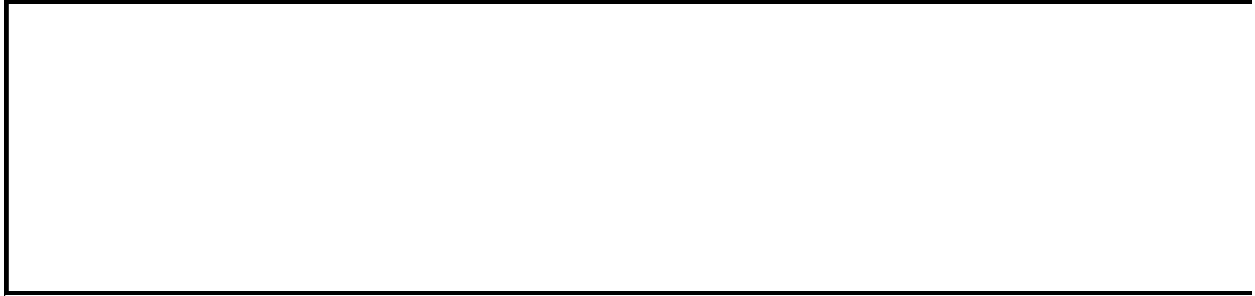
*expression* Required. An expression that returns a **ChChart** object.



## Example

This example sets the angle of the first pie-chart or doughnut-chart slice to 45 degrees. `Charts(0)` must refer to a pie or doughnut chart.

```
ChartSpace1.Charts(0).FirstSliceAngle = 45
```



# Floor Property

-

Returns a **ChSurface** object that represents the floor of a three-dimensional chart. Use the properties and methods of the returned **ChSurface** object to format the floor of the specified chart.

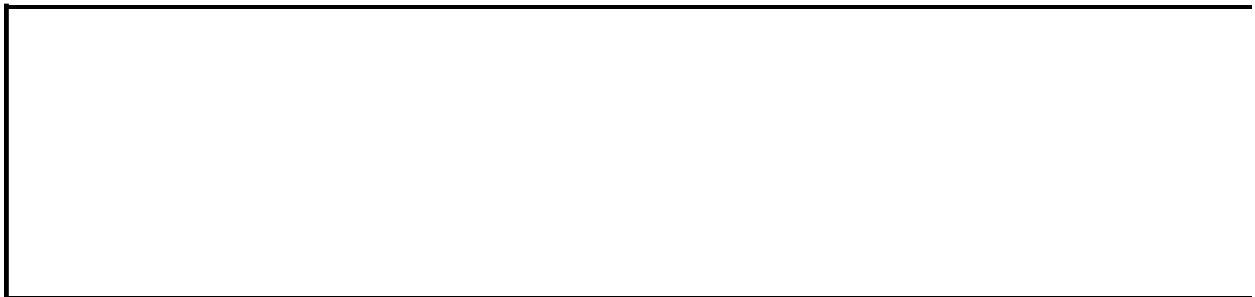
*expression*.**Floor**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then formats the back wall, side wall, and floor of the chart.

```
Sub FormatWallsFloor()  
  
    Dim cht3DColumn  
    Dim chConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Column chart.  
    cht3DColumn.Type = chConstants.chChartTypeColumnClustered3D  
  
    ' Format the back wall of the chart.  
    cht3DColumn.BackWall.Interior.SetSolid "Yellow"  
    cht3DColumn.BackWall.Thickness = 5  
  
    ' Format the side wall of the chart.  
    cht3DColumn.SideWall.Interior.SetSolid "Yellow"  
    cht3DColumn.SideWall.Thickness = 5  
  
    ' Format the floor of the chart.  
    cht3DColumn.Floor.Interior.SetSolid "Blue"  
    cht3DColumn.Floor.Thickness = 5  
  
End Sub
```



# Font Property

-  
Returns a [ChFont](#), [Font](#), or [PivotFont](#) object that represents the font for the specified object (the returned object type depends on the object to which this property is applied).

*expression*.**Font**

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

## Example

This example sets the number format and font for the active cell.

```
Sub SetFont()  
    Dim rngCurrentCell  
    Set rngCurrentCell = Spreadsheet1.ActiveCell  
    rngCurrentCell.NumberFormat = "0.###"  
    rngCurrentCell.Font.Color = "Blue"  
    rngCurrentCell.Font.Name = "Times New Roman"  
End Sub
```



# FontName Property

-

Returns or sets the name of the font in the specified **RecordNavigationControl** object. Read/write **String**.

*expression*.**FontName**

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

## Example

This example sets the font name to "Courier New" for control RNC1.

```
RNC1.FontName = "Courier New"
```



# ForeColor Property

Returns or sets the foreground color for the specified object or area. When you set this property, you can use either a **Long** value representing a red-green-blue (RGB) color value or a **String** value representing a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create an RGB color value (for example, red is RGB(255, 0, 0)). Read/write **Variant**.

*expression*.**ForeColor**

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

## Remarks

This property always returns the color as a **Long** value representing an RGB color value.



## Example

This example sets the font size, foreground color, and background color for the title bar in PivotTable1 .

```
Sub Format_Titlebar()  
    Dim vwView  
  
    Set vwView = PivotTable1.ActiveView  
  
    ' Set the background color of the title bar.  
    vwView.Label.BackColor = "DarkSalmon"  
  
    ' Set the font size of the title bar.  
    vwView.Label.Font.Size = 16  
  
    ' Set the foreground color of the title bar.  
    vwView.Label.ForeColor = "Sienna"  
End Sub
```



# Format Property

-

Returns or sets a **String** that represents the number formatting for the specified element. Read/write.

*expression*.**Format**

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

---

---

# FormatMap Property

-

Returns a **ChFormatMap** object that represents the format map for the specified series.

*expression*.**FormatMap**

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



# Formula Property

-

Returns or sets the object's formula in A1-style notation and in the language of the script. If the cell contains a constant, this property returns the constant. If the cell is empty, it returns an empty string. If the cell contains a formula, the **Formula** property returns the formula as a string in the same format that it would be displayed if the formula were being actively edited (including the equal sign).

If you set the value or formula of a cell to a date, the Spreadsheet component checks to see whether that cell is already formatted with one of the date or time number formats. If not, the Spreadsheet component 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 an array of the same dimensions. Similarly, you can put the formula into an array.

Setting the formula for a multiple-cell range fills all cells in the range with the formula. Read/write **Variant**.

*expression*.**Formula**

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

## Example

This example sets the formula, column width, and number format for cell B5 on Sheet1 in Spreadsheet1.

```
Sub SetFormula()  
    Dim rngRandomNum  
  
    ' Set a variable to cell B5 on Sheet1.  
    Set rngRandomNum =  
Spreadsheet1.Worksheets("Sheet1").Range("B5")  
  
    ' Insert a formula into cell B5.  
    rngRandomNum.Formula = "=5*RAND()"  
  
    ' Set the number format for cell B5.  
    rngRandomNum.NumberFormat = "#.###"  
  
    ' Autofit column B.  
    rngRandomNum.Columns.AutoFit  
  
End Sub
```



# FormulaArray Property

-

Returns or sets a **Variant** representing 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 a null string ("").  
Read/write.

*expression*.**FormulaArray**

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

## Example

This example enters the array formula =SUM(A1:A3) in cells E1:E3 on the active worksheet.

```
Spreadsheet1.ActiveSheet.Range("E1:E3").FormulaArray = _  
    "=SUM(A1:A3)"
```



# FormulaLocal Property

-

Returns or sets the range formula in the language that the user is working in. If the cell contains a constant, this property returns the constant. If the cell is empty, it returns an empty string. If the cell contains a formula, the **Formula** property returns the formula as a string in the same format that it would be displayed if the formula were being actively edited (including the equal sign).

If you set the value or formula of a cell to a date, the Spreadsheet component checks to see whether that cell is already formatted with one of the date or time number formats. If not, the Spreadsheet component 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 an array of the same dimensions. Similarly, you can put the formula into an array.

Setting the formula for a multiple-cell range fills all cells in the range with the formula. Read/write **Variant**.

*expression*.**FormulaLocal**

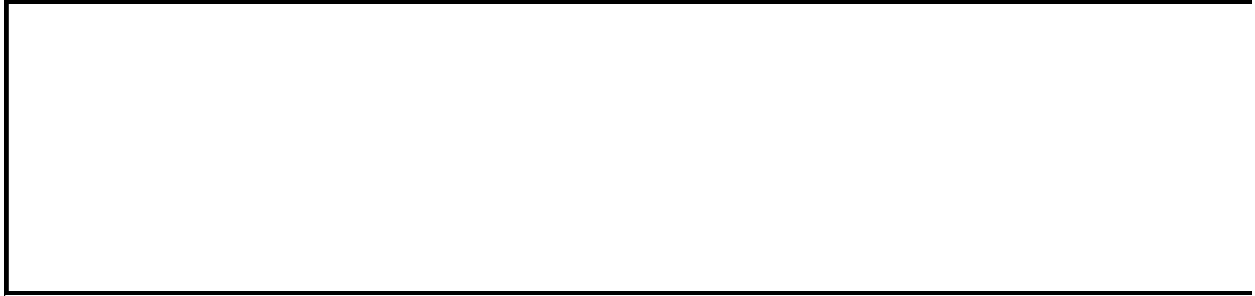
*expression* Required. An expression that returns a **Range** object.



## Example

This example displays the formula in the active cell of Spreadsheet1 in the language of the user.

```
MsgBox Spreadsheet1.ActiveCell.FormulaLocal
```



# FreezePanes Property

-

Set this property to **True** to create panes in the active worksheet. The panes are created above and to the left of the active cell. For example, setting this property to **True** while cell D5 is the active cell results in a horizontal pane between rows 4 and 5 and a vertical pane between columns C and D. Set this property to **False** to remove all panes from a worksheet. Read/write **Boolean**.

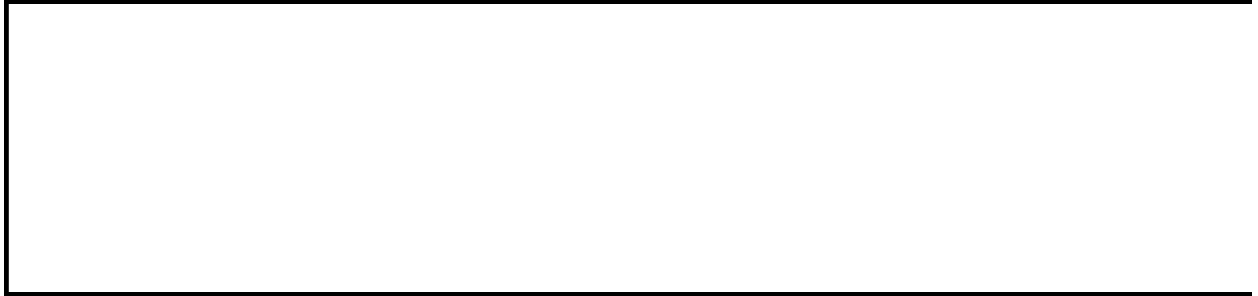
*expression*.**FreezePanes**

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

## Example

This example creates panes in the active worksheet.

```
Spreadsheet1.ActiveWindow.FreezePanes = True
```



▼ [Show All](#)

# Function Property

Returns or sets the function to be used for generating aggregate values.  
Read/write [PivotTotalFunctionEnum](#).

PivotTotalFunctionEnum can be one of these PivotTotalFunctionEnum constants.

**plFunctionAverage**

**plFunctionCalculated**

**plFunctionCount**

**plFunctionMax**

**plFunctionMin**

**plFunctionStdDev**

**plFunctionStdDevP**

**plFunctionSum**

**plFunctionUnknown**

**plFunctionVar**

**plFunctionVarP**

*expression*.**Function**

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

## Remarks

The **Sum** operator is the default value for numeric data types. The **Count** operator is the default value for other data types.

# GapDepth Property

-

Returns or sets a **Long** specifying the amount of spacing along the z-axis for adjacent data series in a three-dimensional chart. Valid values range from 0 to 500. Read/write.

*expression*.**GapDepth**

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

## Remarks

Setting this property to a value of 100 results in the gaps that are the same thickness as the data series series.



## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then sets the gap depth of the chart.

```
Sub SetGapDepth()  
    Dim cht3DColumn As ChChart  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Column chart.  
    cht3DColumn.Type = chChartTypeColumn3D  
  
    ' Set the gap depth.  
    cht3DColumn.GapDepth = 75  
End Sub
```



# GapWidth Property

-

Returns or sets the amount of spacing between markers in adjacent categories, as a percentage of the column width. A value of zero provides no space between adjacent category markers, while positive values create a gap. Read/write **Long**.

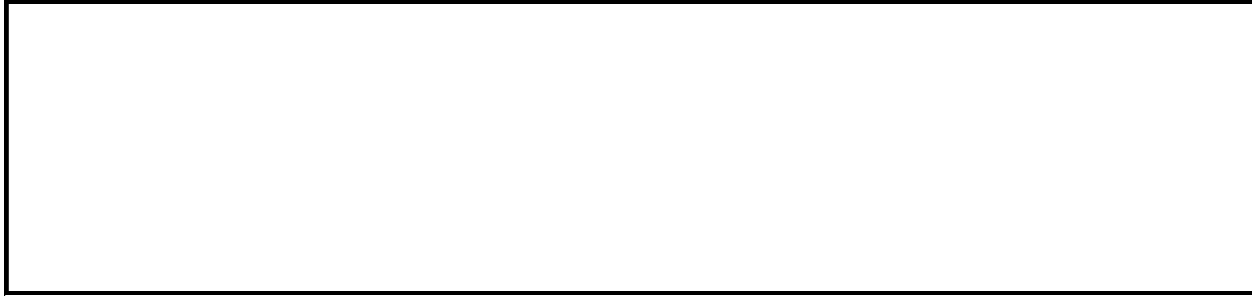
*expression*.**GapWidth**

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

## Example

This example sets the spacing between markers in adjacent categories to 250 points.

```
ChartSpace1.Charts(0).GapWidth = 250
```



# GradientDegree Property

-

Returns a **Double** indicating the gradient degree of the specified shaded fill as a value from 0.0 (dark) through 1.0 (light). Read-only.

*expression*.**GradientDegree**

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

## Remarks

Use the **SetOneColorGradient** method to set the gradient degree for a **ChInterior** object.

▼ [Show All](#)

# GradientStyle Property

-

Returns a [ChartGradientStyleEnum](#) constant indicating the gradient style for the specified **ChInterior** object. Read-only.

ChartGradientStyleEnum can be one of these ChartGradientStyleEnum constants.

**chGradientDiagonalDown**

**chGradientDiagonalUp**

**chGradientFromCenter**

**chGradientFromCorner**

**chGradientHorizontal**

**chGradientVertical**

*expression*.**GradientStyle**

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

## Remarks

Use the **SetPresetGradient**, **SetOneColorGradient** or **SetTwoColorGradient** method to set the gradient style for a **ChInterior** object.



▼ [Show All](#)

# GradientVariant Property

-

Returns a [ChartGradientVariantEnum](#) constant indicating the shade variant for the specified **ChInterior** object. Read-only.

ChartGradientVariantEnum can be one of these ChartGradientVariantEnum constants.

**chGradientVariantCenter**

**chGradientVariantEdges**

**chGradientVariantEnd**

**chGradientVariantStart**

*expression*.**GradientVariant**

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

## Remarks

Use the **OneColorGradient** or **TwoColorGradient** method to set the gradient variant for a **ChInterior** object.

# GridlineColor Property

Returns or sets the gridline color as an RGB value. Read/write **Long**.

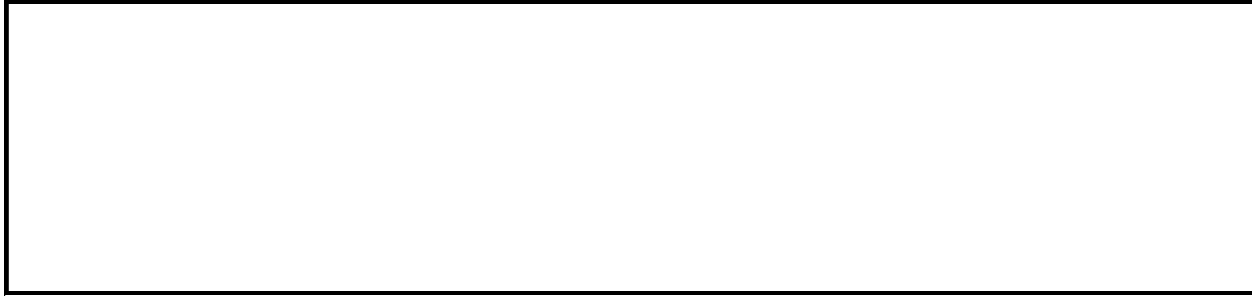
*expression*.**GridlineColor**

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

## Example

The following example sets the gridline color of the active window in Spreadsheet1 to Red.

```
Spreadsheet1.ActiveWindow.GridlineColor = RGB(255,0,0)
```



▼ [Show All](#)

# GridlineColorIndex Property

-  
Returns or sets the gridline color as an index into the current color palette, or as an [XLColorIndex](#) constant. Read/write.

XLColorIndex can be one of these XLColorIndex constants.

**xlColorIndexAutomatic**

**xlColorIndexNone**

























































*expression*.**GridlineColorIndex**

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

## Remarks

Set this property to **xlColorIndexAutomatic** to specify the automatic color.

The following illustration shows the color-index values in the default color palette.

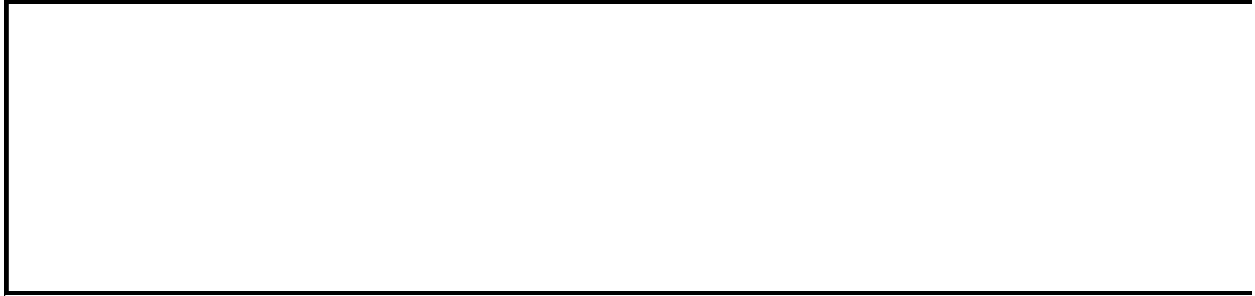
1		2		3		4		5		6		7	
8		9		10		11		12		13		14	
15		16		17		18		19		20		21	
22		23		24		25		26		27		28	
29		30		31		32		33		34		35	
36		37		38		39		40		41		42	
43		44		45		46		47		48		49	
50		51		52		53		54		55		56	



## Example

This example sets the gridline color in the active window of Spreadsheet1 to blue.

```
Spreadsheet1.ActiveWindow.GridlineColorIndex = 5
```



# GridX Property

-

Returns or sets the number of dotted gridlines per inch on the x-axis in the specified data access page's designer default section. Read/write **Long**.

*expression*.**GridX**

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

--

# GridY Property

-

Returns or sets the number of dotted gridlines per inch on the y-axis in the specified data access page's designer default section. Read/write **Long**.

*expression*.**GridY**

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



# GroupedAutoFit Property

-

**True** if the specified field's column width is set automatically when the field is used on the row axis or the column axis. To set the width of a field's column, set the property to **False**. Then, set the **GroupedWidth** property of the field to the desired width. The default value is **True**. Read/write **Boolean**.

---

*expression*.**GroupedAutoFit**

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

## Example

This example disables the **GroupedAutoFit** property of the "ProductName" field in PivotTable1 and then sets the width of the field to 150 pixels.

```
Sub Set_ColumnWidth()  
    Dim fldProducts  
  
    ' Set a variable to the ProductName field.  
    Set fldProducts = PivotTable1.ActiveView.FieldSets("ProductName")  
  
    ' Set the GroupedAutoFit for the ProductName field.  
    fldProducts.GroupedAutoFit = False  
  
    ' Set the width of the ProductName field to 150 pixels.  
    fldProducts.GroupedWidth = 150  
End Sub
```

# GroupedBackColor Property

-

Returns or sets a **Variant** representing the background color of a field when it has been grouped. Read/write.

*expression*.**GroupedBackColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is `RGB(255, 0, 0)`).

## Example

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

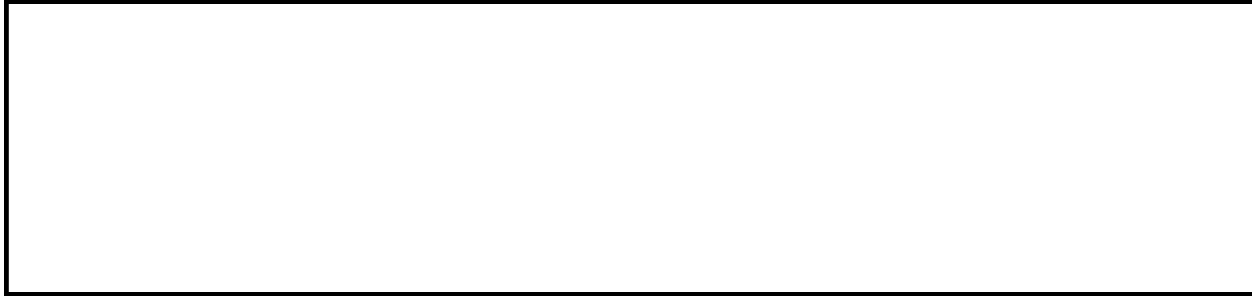
```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
    ' Set the height for the field when it is grouped.  
    pfGroupedField.GroupedHeight = 15
```



```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



# GroupedFont Property

-

Returns a **PivotFont** object that represents the font for the specified field when it has been grouped.

*expression*.**GroupedFont**

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

## Example

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

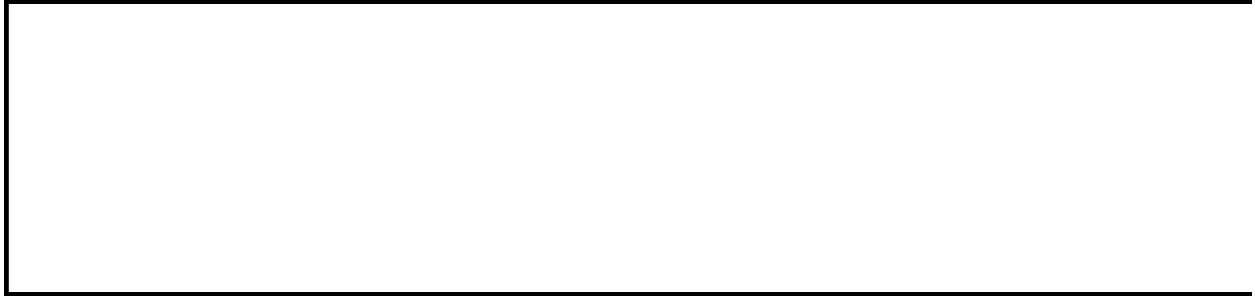
```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
    ' Set the height for the field when it is grouped.  
    pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



# GroupedForeColor Property

-

Returns or sets a **Variant** representing the foreground color of a field when it has been grouped. Use this property to set the color of a grouped item. Read/write.

*expression*.**GroupedForeColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

## Example

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

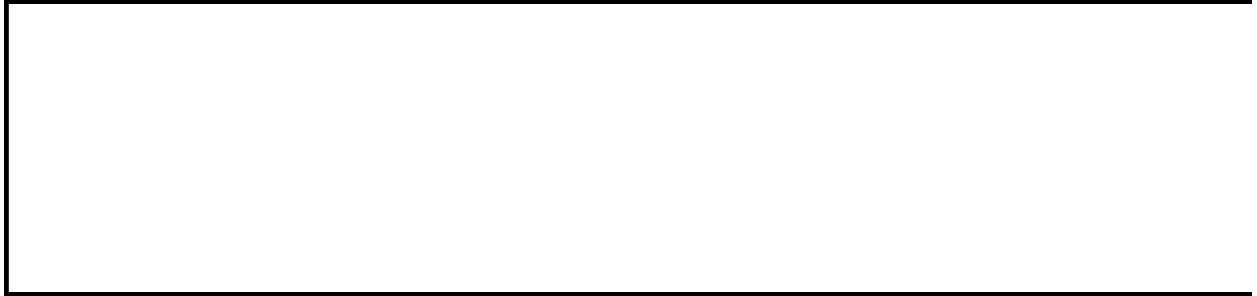
```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
    ' Set the height for the field when it is grouped.  
    pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```





▼ [Show All](#)

# GroupedHAlignment Property

-

Returns or sets a [PivotHAlignmentEnum](#) constant that represents the horizontal alignment of the specified field when it has been grouped. Read/write.

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

*expression*.**GroupedHAlignment**

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

## Example

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

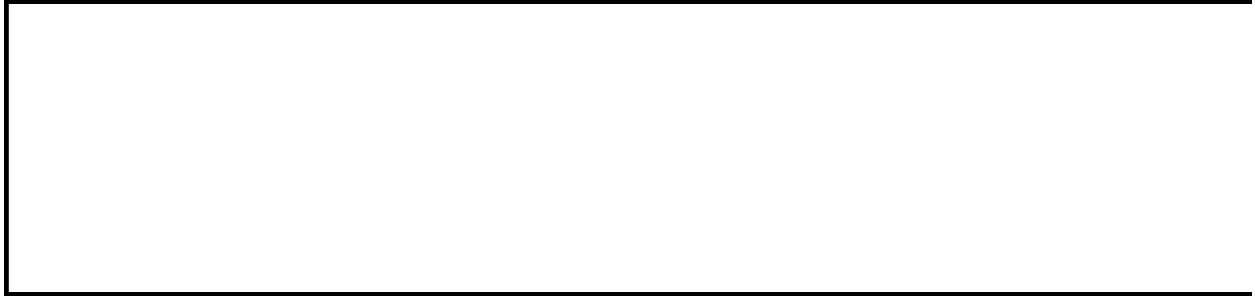
```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
    ' Set the height for the field when it is grouped.  
    pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



# GroupedHeight Property

-  
Returns or sets a **Long** that represents the height of a field when it has been grouped. Read/write.

*expression*.**GroupedHeight**

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

## Example

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

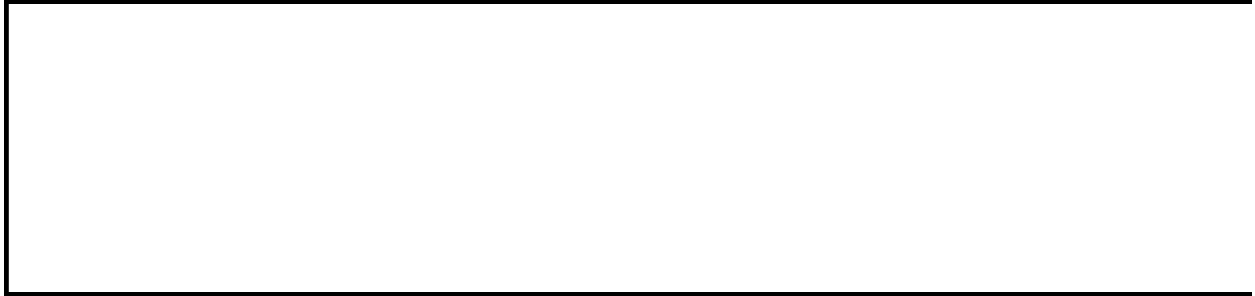
```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
    ' Set the height for the field when it is grouped.  
    pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



# GroupedWidth Property

-

Returns or sets the width of the specified field (pixels) when the field is displayed on the row axis or the column axis. The default value is 50 pixels. This property is ignored if the **GroupedAutoFit** property for the specified field is set to **True**. Read/write **Long**.

*expression*.**GroupedWidth**

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

---

---



## Example

This example disables the **GroupedAutoFit** property of the "ProductName" field in PivotTable1 and then sets the width of the field to 150 pixels.

```
Sub Set_ColumnWidth()  
    Dim fldProducts  
  
    ' Set a variable to the ProductName field.  
    Set fldProducts = PivotTable1.ActiveView.FieldSets("ProductName")  
  
    ' Set the GroupedAutoFit for the ProductName field.  
    fldProducts.GroupedAutoFit = False  
  
    ' Set the width of the ProductName field to 150 pixels.  
    fldProducts.GroupedWidth = 150  
End Sub
```

# GroupEnd Property

-

Returns or sets a **Variant** representing the ending value of the grouping interval for the specified field. Read/write.

*expression*.**GroupEnd**

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

## Remarks

If the specified field contains values greater than the setting for this property, then a group titled " $\geq(\textit{GroupEnd} + 1)$ " is created where *GroupEnd* is the value specified for this property. This group will contain all values that are greater than the setting for this property.

If this property is not set or is set to **Empty**, then the largest value in the field is used as the ending value.

If the current setting for the **GroupOn** property is **plGroupOnPrefixChars**, then you will receive a run-time error when you set this property.

## Example

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

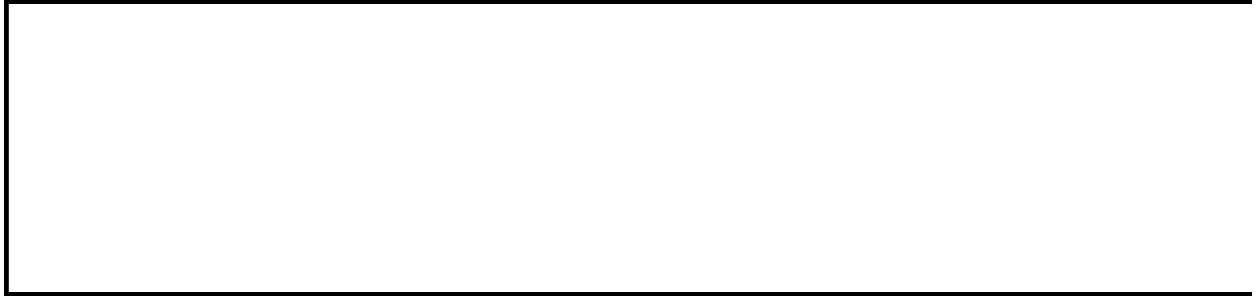
```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
    ' Set the height for the field when it is grouped.  
    pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



# GroupField Property

-

Returns a **PivotResultGroupField** object. Use the properties of the returned object to access the source field and axis properties of the specified axis member.

*expression*.**GroupField**

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

--

# GroupFields Property

Returns the **PivotResultGroupFields** collection for the specified group axis.

*expression*.**GroupFields**

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



# GroupFilterControl Property

-  
Returns or sets the ID of the list box or combo box used to set currency.  
Read/write **String**.

*expression*.**GroupFilterControl**

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

---

---



# GroupFilterField Property

-

Returns or sets the filter string used by the group filter control (the field on which the control will set currency). This string must be the name of a page field in the group level's record source. Read/write **String**.

*expression*.**GroupFilterField**

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

---

---

# GroupFooter Property

-

**True** if the specified section has a footer. You can set this property for any banding level except the innermost one. The default value is **False**. Read/write **Boolean**.

*expression*.**GroupFooter**

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

# GroupHeader Property

-  
**True** if the specified section has a header. The default value is **True**. Read/write **Boolean**.

*expression*.**GroupHeader**

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

--

# GroupingDefs Property

-

Returns the [GroupingDefs](#) collection for the specified recordset definition. This collection contains the [GroupingDef](#) objects that create grouping levels for the detail records.

*expression*.**GroupingDefs**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

▼ [Show All](#)

# GroupingTotalFunction Property

Returns or sets a [ChartGroupingTotalFunctionEnum](#) constant that represents the function used to display the values in a group. Read/write.

ChartGroupingTotalFunctionEnum can be one of these ChartGroupingTotalFunctionEnum constants.

**chFunctionAvg**

**chFunctionCount**

**chFunctionDefault**

**chFunctionMax**

**chFunctionMin**

**chFunctionSum**

*expression*.**GroupingTotalFunction**

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

## Example

This example converts the first chart in Chartspace1 to a line chart, then formats the category axis so that the values are grouped by month. The average value of each month is displayed on the chart.

```
Sub FormatTimeScaling()

    Dim chConstants
    Dim axCategory

    Set chConstants = ChartSpace1.Constants

    ' Change the chart to a Line chart.
    ChartSpace1.Charts(0).Type = chConstants.chChartTypeLine

    ' Set a variable to the category axis.
    Set axCategory = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo

    ' Specify that you will determine the grouping settings of the
    ' axis. Note that this line of code is necessary only if the
    ' GroupingType property for the axis has been previously set to
    ' chAxisGroupingNone.
    axCategory.GroupingType = chConstants.chAxisGroupingManual

    ' Group the category axis by month.
    axCategory.GroupingUnitType = chConstants.chAxisUnitMonth

    ' Create a new grouping for every month.
    axCategory.GroupingUnit = 1

    ' Display the average of the items in each group.
    axCategory.GroupingTotalFunction = chConstants.chFunctionAvg

    ' A tick label is displayed for every month.
    axCategory.TickLabelUnitType = chConstants.chAxisUnitMonth

    ' A tick mark is displayed for every three months.
    axCategory.TickMarkUnitType = chConstants.chAxisUnitQuarter

End Sub
```





▼ [Show All](#)

# GroupingType Property

-

Returns or sets a [ChartAxisGroupingEnum](#) constant that represents whether or not the items on a chart axis are grouped, and if so, whether the grouping was done automatically. Read/write.

ChartAxisGroupingEnum can be one of these ChartAxisGroupingEnum constants.

**chAxisGroupingAuto** The **Chart** control chooses the grouping unit and type.

**chAxisGroupingManual** You will use the **GroupingUnitType** and **GroupingUnit** properties to perform the grouping.

**chAxisGroupingNone** Do not group the axis.

*expression*.**GroupingType**

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

## Remarks

The **Chart** control automatically creates a time scale and groups on the category axis when the following conditions are true:

- The **Chart** control detects that the category information is a date.
- The **Chart** control is bound to a PivotTable list, and the PivotTable list is not bound to an OLAP data source.

## Example

This example disables time scaling on the category axis of the first chart in Chartspace1.

```
Sub DisableTimeScaling()  
    Dim chConstants  
    Dim axCategory  
  
    Set chConstants = ChartSpace1.Constants  
    ' Set a variable to the category axis.  
    Set axCategory = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo  
  
    ' Disable time scaling on the category axis.  
    axCategory.GroupingType = chConstants.chAxisGroupingNone  
End Sub
```



# GroupingUnit Property

-

Returns or sets a **Long** value that represents the number of items in a group.  
Read/write.

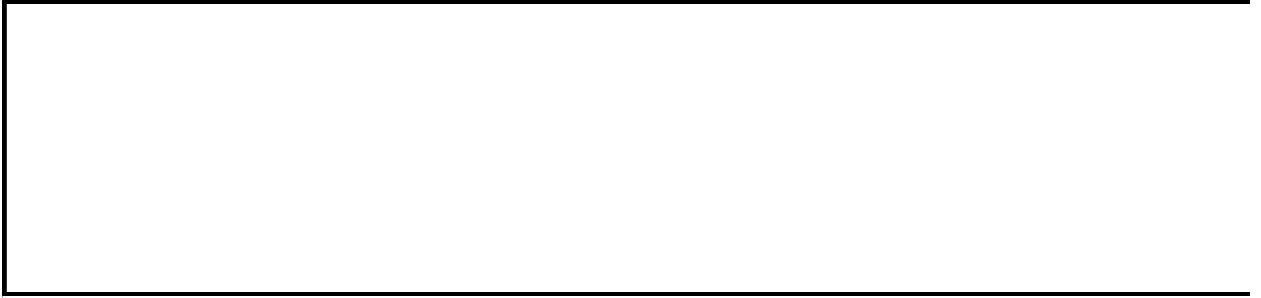
*expression*.**GroupingUnit**

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

## Example

This example converts the first chart in Chartspace1 to a line chart, then formats the category axis so that the values are grouped by month. The average value of each month is displayed on the chart.

```
Sub FormatTimeScaling()  
  
    Dim chConstants  
    Dim axCategory  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Change the chart to a Line chart.  
    ChartSpace1.Charts(0).Type = chConstants.chChartTypeLine  
  
    ' Set a variable to the category axis.  
    Set axCategory = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo  
  
    ' Specify that you will determine the grouping settings of the  
    ' axis. Note that this line of code is necessary only if the  
    ' GroupingType property for the axis has been previously set to  
    ' chAxisGroupingNone.  
    axCategory.GroupingType = chConstants.chAxisGroupingManual  
  
    ' Group the category axis by month.  
    axCategory.GroupingUnitType = chConstants.chAxisUnitMonth  
  
    ' Create a new grouping for every month.  
    axCategory.GroupingUnit = 1  
  
    ' Display the average of the items in each group.  
    axCategory.GroupingTotalFunction = chConstants.chFunctionAvg  
  
    ' A tick label is displayed for every month.  
    axCategory.TickLabelUnitType = chConstants.chAxisUnitMonth  
  
    ' A tick mark is displayed for every three months.  
    axCategory.TickMarkUnitType = chConstants.chAxisUnitQuarter  
  
End Sub
```





▼ [Show All](#)

# GroupingUnitType Property

-

Returns or sets a [ChartAxisUnitTypeEnum](#) constant that represents how items are grouped on an axis.

ChartAxisUnitTypeEnum can be one of these ChartAxisUnitTypeEnum constants.

**chAxisUnitDay**

**chAxisUnitMonth**

**chAxisUnitQuarter**

**chAxisUnitWeek**

**chAxisUnitYear**

*expression*.**GroupingUnitType**

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

## Remarks

Setting this property sets the **GroupingType** property to **chAxisGroupingManual**.

The **Chart** control automatically creates a time scale and groups on the category axis when the following conditions are true:

- The **Chart** control detects that the category information is a date.
- The **Chart** control is bound to a PivotTable list, and the PivotTable list is not bound to an OLAP data source.

## Example

This example converts the first chart in Chartspace1 to a line chart, then formats the category axis so that the values are grouped by month. The average value of each month is displayed on the chart.

```
Sub FormatTimeScaling()

    Dim chConstants
    Dim axCategory

    Set chConstants = ChartSpace1.Constants

    ' Change the chart to a Line chart.
    ChartSpace1.Charts(0).Type = chConstants.chChartTypeLine

    ' Set a variable to the category axis.
    Set axCategory = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo

    ' Specify that you will determine the grouping settings of the
    ' axis. Note that this line of code is necessary only if the
    ' GroupingType property for the axis has been previously set to
    ' chAxisGroupingNone.
    axCategory.GroupingType = chConstants.chAxisGroupingManual

    ' Group the category axis by month.
    axCategory.GroupingUnitType = chConstants.chAxisUnitMonth

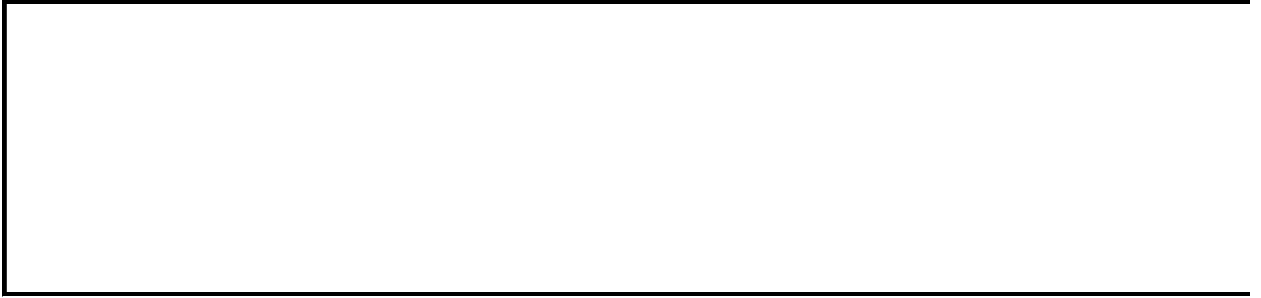
    ' Create a new grouping for every month.
    axCategory.GroupingUnit = 1

    ' Display the average of the items in each group.
    axCategory.GroupingTotalFunction = chConstants.chFunctionAvg

    ' A tick label is displayed for every month.
    axCategory.TickLabelUnitType = chConstants.chAxisUnitMonth

    ' A tick mark is displayed for every three months.
    axCategory.TickMarkUnitType = chConstants.chAxisUnitQuarter

End Sub
```



▼ [Show All](#)

# GroupInterval Property

► [GroupInterval property as it applies to the \*\*PivotField\*\* object.](#)

Returns or sets a **Double** that represents the the group interval for a field. This property is interpreted based upon the current setting for the **GroupOn** property. The following table describes how this property is interpreted based on the current **GroupOn** setting. Read/write.

GroupOn setting	Meaning of GroupInterval
<b>plGroupOnEachValue</b>	Ignored.
<b>plGroupOnPrefixChars</b>	Number of initial characters to use from each value.
<b>plGroupOnYears</b>	Number of years in each group.
<b>plGroupOnQtrs</b>	Number of quarters in each group.
<b>plGroupOnMonths</b>	Number of months in each group.
<b>plGroupOnWeeks</b>	Number of weeks in each group.
<b>plGroupOnDays</b>	Number of days in each group.
<b>plGroupOnHours</b>	Number of hours in each group.
<b>plGroupOnMinutes</b>	Number of minutes in each group.
<b>plGroupOnSeconds</b>	Number of seconds in each group.
<b>plGroupOnInterval</b>	Numeric size of each group.

*expression*.**GroupInterval**

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

► [GroupInterval property as it applies to the \*\*GroupLevel\*\* and \*\*PageField\*\* objects.](#)

Returns or sets a **Double** that represents the group interval. Read/write.

*expression*.**GroupInterval**

*expression* Required. An expression that returns one of the above objects.



## Example

► [As it applies to the \*\*PivotField\*\* object.](#)

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

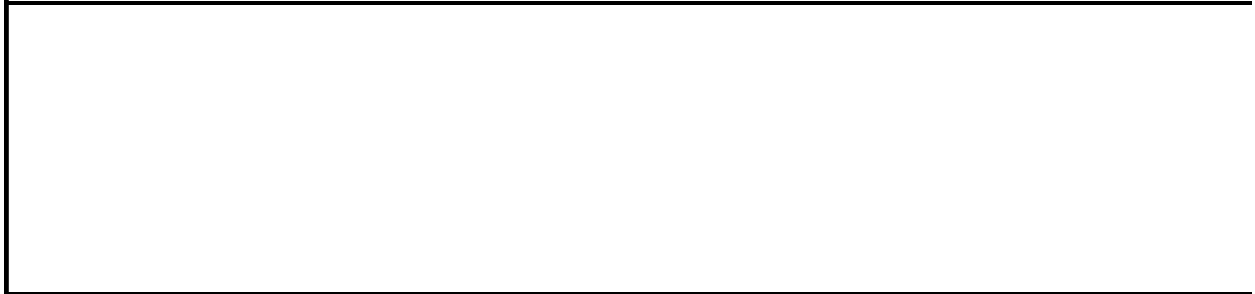
```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
' Set the height for the field when it is grouped.  
pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



# GroupLevel Property

-

Returns the [GroupLevel](#) object for the specified data page. A **GroupLevel** object corresponds to a recordset produced by a recordset definition or grouping definition. Read-only.

---

*expression*.**GroupLevel**

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

# GroupLevels Property

-  
Returns the [GroupLevels](#) collection for the specified data source control. Read-only.

*expression*.**GroupLevels**

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

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).



▼ [Show All](#)

# GroupOn Property

► [GroupOn property as it applies to the \*\*PivotField\*\* object.](#)

Returns or sets a [PivotFieldGroupOnEnum](#) constant that represents the grouping settings for a field. Use this property to determine how items are grouped in a PivotTable field. Read/write.

PivotFieldGroupOnEnum can be one of these PivotFieldGroupOnEnum constants.

**plGroupOnEachValue** Specifies no grouping. The default setting.

**plGroupOnPrefixChars** Group text values by the first *N* characters of each value, where *N* is the **GroupInterval** property value. This setting is valid only for textual fields.

**plGroupOnYears** Group date or time values by years.

**plGroupOnQtrs** Group date or time values by quarters.

**plGroupOnMonths** Group date or time values by month.

**plGroupOnWeeks** Group date or time values by week.

**plGroupOnDays** Group date or time values by their respective day number

**plGroupOnHours** Group date or time values by their respective hour number.

**plGroupOnMinutes** Group date or time values by their respective hour number.

**plGroupOnSeconds** Group date or time values by their respective second number.

**plGroupOnInterval** Group numeric values by the interval specified in the **GroupInterval** property.

*expression*.**GroupOn**

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

## Remarks

Once you have set the **GroupOn** property, you can use the **GroupInterval**, **GroupStart**, and **GroupEnd** properties to further define how to group items in a field.

Setting this property affects the way in which values in each field are displayed on the row, column, and filter areas of a PivotTable list. The settings for this property do not affect how values are displayed in the detail area of a PivotTable list. However, the filter dropdown will reflect the **GroupOn** setting for a field in the detail area of a PivotTable list.

Setting this property of a PivotTable field clears its **IncludedMembers** and **ExcludedMembers** properties if they have been set.

► [GroupOn property as it applies to the \*\*GroupLevel\*\* and \*\*PageField\*\* objects.](#)

Returns or sets a **DscGroupOnEnum** that represents the way **GroupLevel** objects or **PageField** objects are grouped. Read/write.

*expression*.**GroupOn**

*expression* Required. An expression that returns one of the objects listed above.

DscGroupOnEnum can be one of these DscGroupOnEnum constants.

**dscDay dscEachValue dscHour dscInterval dscMinute dscMonth dscPrefix  
dscQuarter dscWeek dscYear**

## Example

► [As it applies to the \*\*PivotField\*\* object.](#)

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

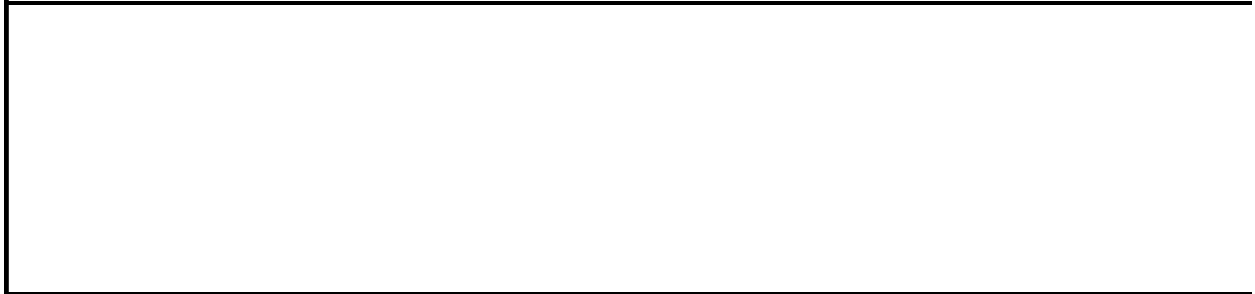
```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```



```
' Set the height for the field when it is grouped.  
pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



# GroupStart Property

-

Returns or sets a **Variant** representing the starting value of the grouping interval for the specified field. Read/write.

*expression*.**GroupStart**

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

## Remarks

If the specified field contains values smaller than the setting for this property, then a group entitled "<*GroupStart* " is created where *GroupStart* is the value specified for this property. This group will contain all values that are smaller than the setting for this property.

If this property is not set or is set to **Empty**, then the smallest value in the field is used as the starting value.

If the current setting for the **GroupOn** property is **plGroupOnPrefixChars**, then you will receive a run-time error when you set this property.

## Example

This example groups the Age field of PivotTable1. Starting at age 15, a new group will be created for every five years until age 80. Then, the font, foreground, background, height and alignment of the resulting groups are formatted.

```
Sub AddGrouping()
```

```
    Dim vwView  
    Dim ptConstants  
    Dim pfGroupedField
```

```
    Set ptConstants = PivotTable1.Constants
```

```
    ' Set a variable to the active view of the PivotTable.  
    Set vwView = PivotTable1.ActiveView
```

```
    ' Set a variable to the Age field.  
    Set pfGroupedField = vwView.FieldSets("Age").Fields("Age")
```

```
    ' Set the GroupOn property so that the Age field will be  
    ' grouped by the GroupInterval setting.  
    pfGroupedField.GroupOn = ptConstants.plGroupOnInterval
```

```
    ' Create a new grouping for every five years.  
    pfGroupedField.GroupInterval = 5
```

```
    ' Start the grouping at age 15.  
    pfGroupedField.GroupStart = 15
```

```
    ' End the grouping at age 80.  
    pfGroupedField.GroupEnd = 80
```

```
    ' Set the font for the field when it is grouped.  
    pfGroupedField.GroupedFont.Bold = True
```

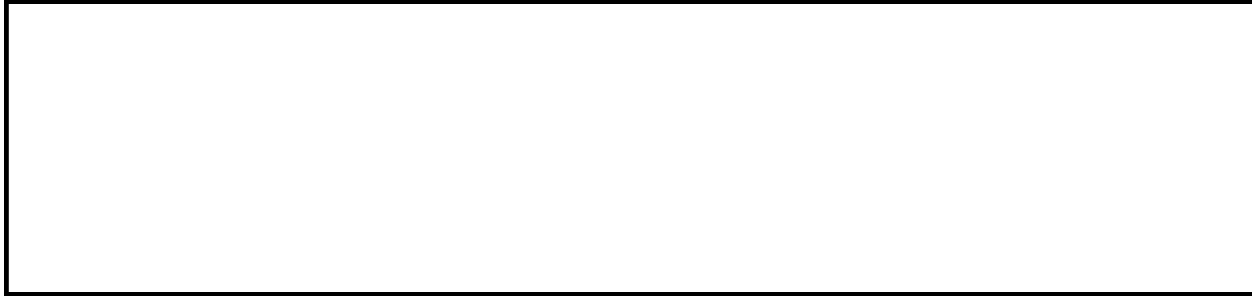
```
    ' Set the foreground color for the field when it is grouped.  
    pfGroupedField.GroupedForeColor = "Black"
```

```
    ' Set the Background color for the field when it is grouped.  
    pfGroupedField.GroupedBackColor = "Blue"
```

```
    ' Set the height for the field when it is grouped.  
    pfGroupedField.GroupedHeight = 15
```

```
' Set the horizontal alignment for the field when it is grouped.  
pfGroupedField.GroupedHAlignment = ptConstants.plHAlignRight
```

```
End Sub
```



▼ [Show All](#)

# HAlignment Property

-

Returns or sets the way data is aligned in the specified label or total. Read/write [PivotHAlignmentEnum](#).

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

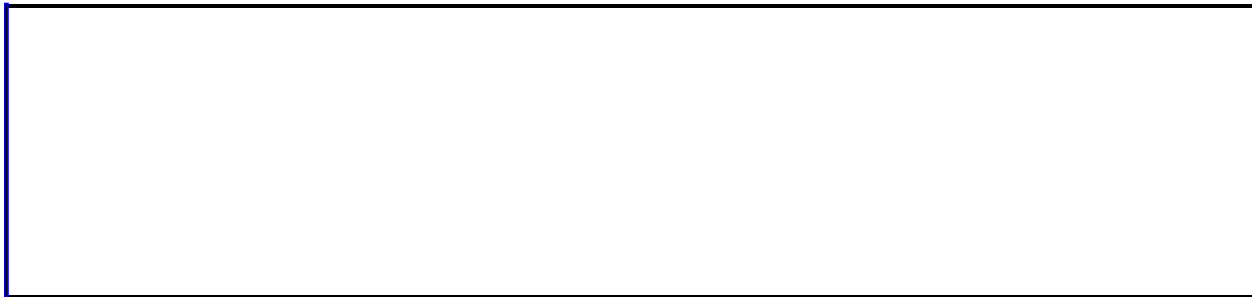
**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

*expression*.**HAlignment**

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



# HasAbsoluteLabels Property

-

Returns or sets whether the legend entries for percentage-calculated segments are displayed as absolute values in the legend. Set this property to **True** in order to display the actual beginning and ending values for a segment that was calculated as a percentage. The default value is **False**. Read/write **Boolean**.

*expression*.**HasAbsoluteLabels**

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



## Example

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created that displays the larger values in the chart with a darker shade of blue.

```
Sub Window_Onload()
```

```
    Dim serSeries1  
    Dim segSegment1 As ChSegment  
    Dim chconstants
```

```
    Set chconstants = ChartSpace1.Constants
```

```
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.
```

```
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=Da  
    ChartSpace1.DataMember = "Order Details"
```

```
    ' The following two lines of code bind Chartspace1 to the Quanti  
    ' in the Order Details table.
```

```
    ChartSpace1.SetData chconstants.chDimCategories, chconstants.chD  
    ChartSpace1.SetData chconstants.chDimValues, chconstants.chDataB
```

```
    ' Create a format map.
```

```
    ChartSpace1.SetData chconstants.chDimFormatValues, chconstants.c
```

```
    ' Set a variable to the first series in the first chart in Chart  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)
```

```
    ' Add a segment to the format map.
```

```
    Set segSegment1 = serSeries1.FormatMap.Segments.Add
```

```
    ' Measure the segment boundaries based upon a percentage.
```

```
    segSegment1.Begin.ValueType = chconstants.chBoundaryValuePercent  
    segSegment1.End.ValueType = chconstants.chBoundaryValuePercent
```

```
    ' Set the beginning value to 0%, and the ending value to 100%.
```

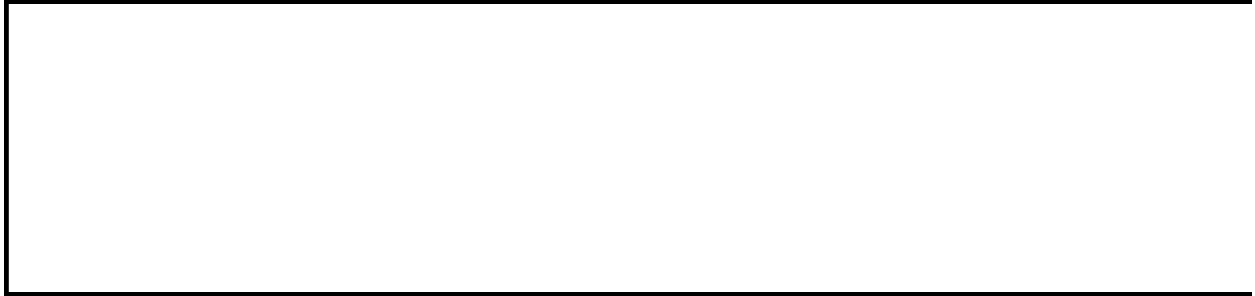
```
    segSegment1.Begin.Value = 0  
    segSegment1.End.Value = 1
```

```
    ' Format the interior of the matching values.
```

```
    segSegment1.Begin.Interior.Color = "White"  
    segSegment1.End.Interior.Color = "Blue"  
    segSegment1.HasAutoDivisions = False
```

```
segSegment1.HasAbsoluteLabels = True  
segSegment1.HasDiscreteDivisions = False
```

```
End Sub
```



# HasArray Property

**True** if the specified cell is part of an array formula. Read-only **Variant**.

*expression*.**HasArray**

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

## Remarks

Use the **CurrentArray** property to determine the the cells that are part of the current array.

## Example

This example determines whether the active cell in Spreadsheet1 is part of an array. If so, the array is selected.

```
If Spreadsheet1.ActiveCell.HasArray Then _  
    Spreadsheet1.ActiveCell.CurrentArray.Select
```



# HasAutoAspectRatio Property

-  
**False** if the aspect ratio of the specified chart has been modified. Set this property to **True** to restore the specified chart to its default aspect ratio. Read-write **Boolean**.

*expression*.**HasAutoAspectRatio**

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

## Example

This example resets the first chart in Chartspace1 to its default aspect ratio.

```
Chartspace1.Charts(0).HasAutoAspectRatio = True
```



# HasAutoCaption Property

-  
**True** if the name for the specified trendline is generated automatically from the trendline type and series index ("Poly. (Series 1)", for example). Setting the trendline's [Caption](#) property sets this property to **False**. Read/write **Boolean**.

*expression*.**HasAutoCaption**

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



## Example

This example sets the caption for the specified trendline caption if the caption is currently set to be generated automatically. Note that `SeriesCollection(0)` must refer to a series that already has a trendline.

```
Sub SetCaption()  
    Dim serZero  
    Dim trndline  
  
    Set serZero = ChartSpace1.Charts(0).SeriesCollection(0)  
    serZero.Line.Color = "red"  
    Set trndline = serZero.Trendlines(0)  
    If trndline.HasAutoCaption Then trndline.Caption = "data trend"  
End Sub
```



# HasAutoChartDepth Property

**False** if the depth of the specified chart has been modified. Set this property to **True** to restore the specified chart to its default depth. Read-write **Boolean**.

*expression*.**HasAutoChartDepth**

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

## Example

This example resets the first chart in Chartspace1 to it's default depth.

```
Chartspace1.Charts(0).HasAutoChartDepth = True
```



# HasAutoDivisions Property

-  
Returns or sets whether divisions are automatically created for the specified segment. The default value is **True**. Read/write **Boolean**.

*expression*.**HasAutoDivisions**

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

## Example

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created that displays the larger values in the chart with a darker shade of blue.

```
Sub Window_Onload()  
  
    Dim serSeries1  
    Dim segSegment1 As ChSegment  
    Dim chconstants  
  
    Set chconstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=Da  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind Chartspace1 to the Quanti  
    ' in the Order Details table.  
    ChartSpace1.SetData chconstants.chDimCategories, chconstants.chD  
    ChartSpace1.SetData chconstants.chDimValues, chconstants.chDataB  
  
    ' Create a format map.  
    ChartSpace1.SetData chconstants.chDimFormatValues, chconstants.c  
  
    ' Set a variable to the first series in the first chart in Chart  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a segment to the format map.  
    Set segSegment1 = serSeries1.FormatMap.Segments.Add  
  
    ' Measure the segment boundaries based upon a percentage.  
    segSegment1.Begin.ValueType = chconstants.chBoundaryValuePercent  
    segSegment1.End.ValueType = chconstants.chBoundaryValuePercent  
  
    ' Set the beginning value to 0%, and the ending value to 100%  
    segSegment1.Begin.Value = 0  
    segSegment1.End.Value = 1  
  
    ' Format the interior of the matching values.  
    segSegment1.Begin.Interior.Color = "White"  
    segSegment1.End.Interior.Color = "Blue"
```

```
segSegment1.HasAutoDivisions = False
```

```
segSegment1.HasAbsoluteLabels = True
```

```
segSegment1.HasDiscreteDivisions = False
```

```
End Sub
```



# HasAutoMajorUnit Property

**True** if the major unit for the specified axis is determined automatically. The default value is **True**. You should use this property only for value axes.  
Read/write **Boolean**.

*expression*.**HasAutoMajorUnit**

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

## Example

This example sets the major unit for the specified axis if the major unit is currently set to be determined automatically.

```
Sub SetMajorUnit()  
    Dim chtConstants  
    Dim axs  
  
    Set chtConstants = ChartSpace1.Constants  
    Set axs = ChartSpace1.Charts(0).Axes(chtConstants.chAxisPosition  
    If axs.HasAutoMajorUnit Then axs.MajorUnit = 5000  
End Sub
```





# HasAutoMaximum Property

**True** if the maximum scale value for the specified axis is set to be determined automatically. The default value is **True**. Read/write **Boolean**.

*expression*.**HasAutoMaximum**

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

## Example

This example sets the maximum scale value for the specified axis if the maximum value is currently set to be determined automatically.

```
Sub SetScale()  
    Dim chConstants  
    Dim axisScale  
  
    Set chConstants = ChartSpace1.Constants  
    Set axisScale = ChartSpace1.Charts(0).Axes(chConstants.chAxisPos  
    If axisScale.HasAutoMaximum Then axisScale.Maximum = 50000  
End Sub
```



# HasAutoMinimum Property

**True** if the minimum scale value for the specified axis is set to be determined automatically. The default value is **True**. Read/write **Boolean**.

*expression*.**HasAutoMinimum**

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

## Example

This example sets the minimum scale value for the specified axis if the minimum value is currently set to be determined automatically.

```
Sub SetScaling()  
    Dim chConstants  
    Dim axisScale  
  
    Set chConstants = ChartSpace1.Constants  
    Set axisScale = ChartSpace1.Charts(0).Axes(chConstants.chAxisPos  
    If axisScale.HasAutoMinimum Then axisScale.Minimum = 10  
End Sub
```



# HasAutoMinorUnit Property

**True** if the minor unit for the specified axis is set to be determined automatically. The default value is **True**. You should use this property only for value axes. Read/write **Boolean**.

*expression*.**HasAutoMinorUnit**

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

## Example

This example sets the minor unit for the specified axis to increments of 500 if the unit is currently set to be determined automatically.

```
Sub SetMinorUnit()  
    Dim chConstants  
    Dim axs  
  
    Set chConstants = ChartSpace1.Constants  
    Set axs = ChartSpace1.Charts(0).Axes(chConstants.chAxisPositionV  
    If axs.HasAutoMinorUnit Then axs.MinorUnit = 500  
End Sub
```



# HasBubbleSize Property

**True** if every data label for the specified series or chart currently displays its bubble size. The default value is **False**. Note that this property is available only for bubble charts. Read/write **Boolean**.

*expression*.**HasBubbleSize**

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

## Remarks

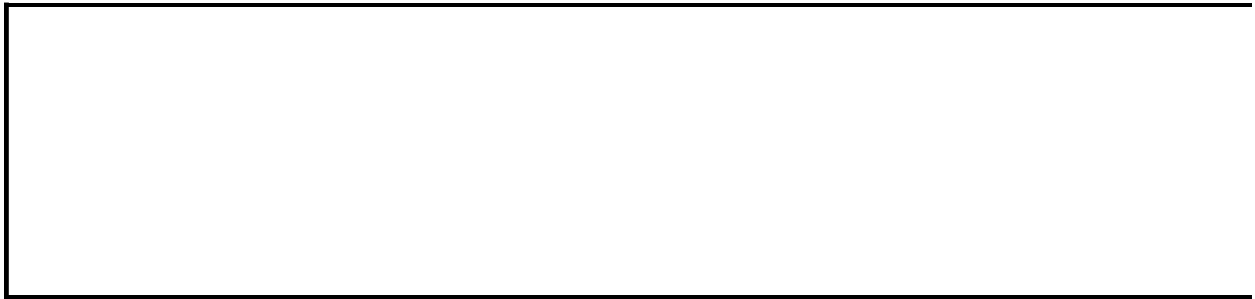
Data label components are always displayed in the following order:  
[SeriesName] [CategoryName] [Value] [BubbleSize] [Percentage].



## Example

This example causes the data labels for the specified series to display their bubble size. Note that `Charts(0)` must refer to a bubble chart.

```
Sub DisplayLabels()  
    Dim dlBubbleLabels  
  
    Set dlBubbleLabels = ChartSpace1.Charts(0).SeriesCollection(0).D  
  
    dlBubbleLabels.HasBubbleSize = True  
End Sub
```



# HasCategoryName Property

**True** if every data label for the specified series or chart currently displays its category name or label. The default value is **False**. Read/write **Boolean**.

*expression*.**HasCategoryName**

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

## Remarks

Data label components are always displayed in the following order:  
[SeriesName] [CategoryName] [Value] [BubbleSize] [Percentage]

## Example

This example causes the data labels for the specified series to display their category and series names.

```
Sub ShowLabels()  
    Dim dl  
  
    Set dl = ChartSpace1.Charts(0).SeriesCollection(0).DataLabelsCol  
  
    ' Display the category names.  
    dl.HasCategoryName = True  
  
    ' Display the series names.  
    dl.HasSeriesName = True  
End Sub
```



# HasChartSpaceLegend Property

**True** if the specified chart workspace has a legend. Read/write **Boolean**.

*expression*.**HasChartSpaceLegend**

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

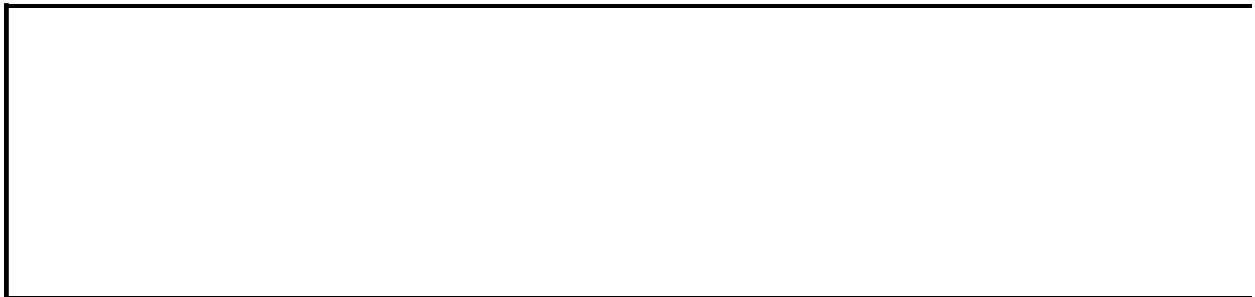
## Remarks

Setting this property to **False** causes the legend to be deleted from the chart workspace. When this happens, all custom formatting is lost and must be reset if the property is subsequently set to **True**.

## Example

This example sets the chart workspace title and positions the chart workspace legend on the left side of the workspace.

```
Sub Format_ChartSpace()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Enable the title for the chartspace.  
    ChartSpace1.HasChartSpaceTitle = True  
  
    ' Set the chartspace title.  
    ChartSpace1.ChartSpaceTitle.Caption = "Monthly Sales Data"  
  
    ' Enable the legend for the chartspace.  
    ChartSpace1.HasChartSpaceLegend = True  
  
    ' Specify the position of the chartspace legend.  
    ChartSpace1.ChartSpaceLegend.Position = chConstants.chLegendPosi  
End Sub
```



# HasChartSpaceTitle Property

**True** if the specified chart workspace has a title. Read/write **Boolean**.

*expression*.**HasChartSpaceTitle**

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



## Remarks

Setting this property to **False** causes the title to be deleted from the chart workspace. When this happens, all custom formatting is lost and must be reset if the property is subsequently set to **True**.

## Example

This example sets the chart workspace title and positions the chart workspace legend on the left side of the workspace.

```
Sub Format_ChartSpace()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Enable the title for the chartspace.  
    ChartSpace1.HasChartSpaceTitle = True  
  
    ' Set the chartspace title.  
    ChartSpace1.ChartSpaceTitle.Caption = "Monthly Sales Data"  
  
    ' Enable the legend for the chartspace.  
    ChartSpace1.HasChartSpaceLegend = True  
  
    ' Specify the position of the chartspace legend.  
    ChartSpace1.ChartSpaceLegend.Position = chConstants.chLegendPosi  
End Sub
```



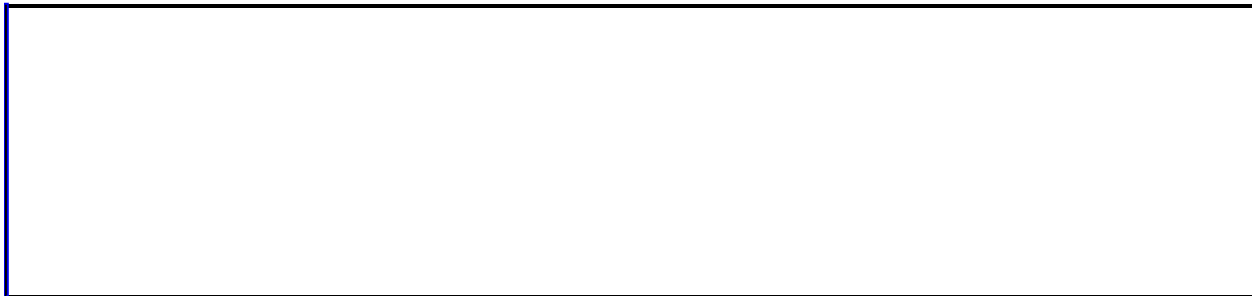
# HasDetails Property

-

**True** if detail records can be displayed for a given cell; **False** if detail records are not available for display. This property is automatically reset whenever the data is requested. If this property is set to **False**, the expansion indicators are not displayed for inner members. This property is always set to **False** if the provider is multidimensional. Read-only **Boolean**.

*expression*.**HasDetails**

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



# HasDiscreteDivisions Property

-

Returns or sets the method used to interpolate the formatting of the specified segment between its beginning and ending values. Setting this property to **True** causes the **Chart** control to assign a number of divisions to the segment. Setting this property to **False** causes the **Chart** control to interpolate between the beginning and ending values of the segment without creating discrete divisions in formatting. The default value is **False**. Read/write **Boolean**.

*expression*.**HasDiscreteDivisions**

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

## Remarks

To illustrate the differences when setting this property to **True** or **False**, assume that you set the following properties for a segment:

- .Begin.Value = 10
- .Begin.Interior.Color = "White"
- .End.Value = 50
- .End.Interior.Color = "Green"

When setting this property to **True**, the **Chart** control would create several divisions which contain a different interpolation of the above color settings. However, points that are very close to each other in value can be formatted very differently because each point is in a different division. A point with a value of 24 may be white while a point with a value of 25 may be a fairly dark shade of green, because a division was created between 24 and 25. Setting this property to **False** results in a more gradual interpolation between white and green.

## Example

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created that displays the larger values in the chart with a darker shade of blue.

```
Sub Window_Onload()
```

```
    Dim serSeries1  
    Dim segSegment1 As ChSegment  
    Dim chconstants
```

```
    Set chconstants = ChartSpace1.Constants
```

```
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.
```

```
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=Da  
    ChartSpace1.DataMember = "Order Details"
```

```
    ' The following two lines of code bind Chartspace1 to the Quanti  
    ' in the Order Details table.
```

```
    ChartSpace1.SetData chconstants.chDimCategories, chconstants.chD  
    ChartSpace1.SetData chconstants.chDimValues, chconstants.chDataB
```

```
    ' Create a format map.
```

```
    ChartSpace1.SetData chconstants.chDimFormatValues, chconstants.c
```

```
    ' Set a variable to the first series in the first chart in Chart  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)
```

```
    ' Add a segment to the format map.
```

```
    Set segSegment1 = serSeries1.FormatMap.Segments.Add
```

```
    ' Measure the segment boundaries based upon a percentage.
```

```
    segSegment1.Begin.ValueType = chconstants.chBoundaryValuePercent  
    segSegment1.End.ValueType = chconstants.chBoundaryValuePercent
```

```
    ' Set the beginning value to 0%, and the ending value to 100%.
```

```
    segSegment1.Begin.Value = 0  
    segSegment1.End.Value = 1
```

```
    ' Format the interior of the matching values.
```

```
    segSegment1.Begin.Interior.Color = "White"  
    segSegment1.End.Interior.Color = "Blue"
```

```
segSegment1.HasAutoDivisions = False
```

```
segSegment1.HasAbsoluteLabels = True
```

```
segSegment1.HasDiscreteDivisions = False
```

```
End Sub
```



# HasFormula Property

-  
**True** if all cells in the range contain formulas, **False** if none contain formulas, and **Null** if some cells contain formulas and others do not. Read-only **Variant**. Use the **IsNull** function to determine if the return value is **Null**.

*expression*.**HasFormula**

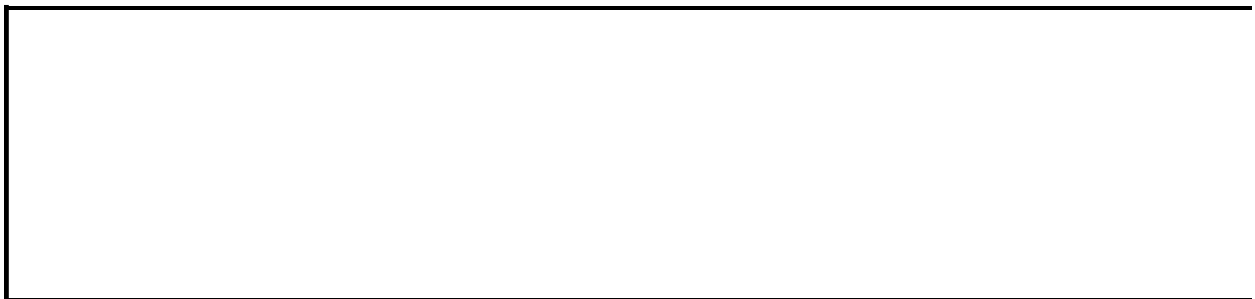
*expression* Required. An expression that returns a **Range** object.



## Example

This example recalculates the active worksheet if any cell in the currently selected range contains a formula.

```
Sub CalcIfSelectionHasFormulas()  
    Dim vntHasFormula  
    Dim rngCurrent  
  
    Set rngCurrent = Spreadsheet1.Selection  
  
    ' Set a variable to the HasFormula property  
    ' for the current selection.  
    vntHasFormula = rngCurrent.HasFormula  
  
    If IsNull(vntHasFormula) Then  
        ' Calculate the active worksheet if the  
        ' selection contains one or more formulas.  
        Spreadsheet1.ActiveSheet.Calculate  
  
    ElseIf vntHasFormula Then  
        ' Calculate the active worksheet if all  
        ' selected cells contain a formula.  
        Spreadsheet1.ActiveSheet.Calculate  
    End If  
End Sub
```



# HasLegend Property

**True** if the specified chart has a legend. Read/write **Boolean**.

*expression*.**HasLegend**

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

## Example

This example sets the specified chart to have a legend.

```
ChartSpace1.Charts(0).HasLegend = True
```



# HasMajorGridlines Property

**True** if the specified axis has major gridlines. Note that any axis can have gridlines. Read/write **Boolean**.

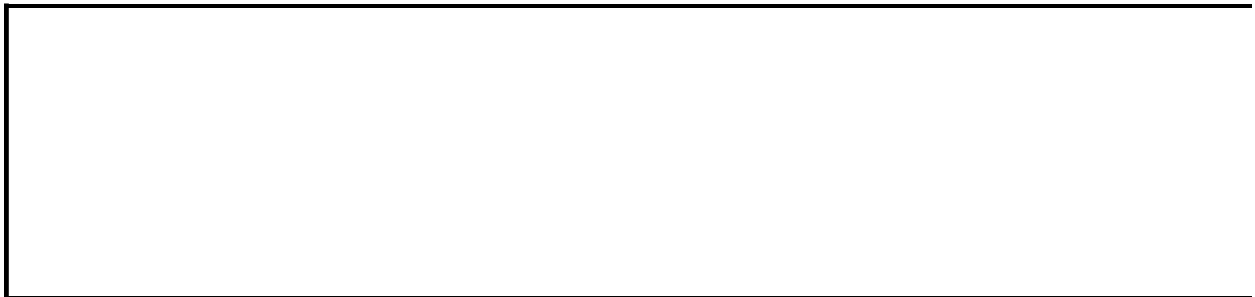
*expression*.**HasMajorGridlines**

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

## Example

This example turns on the major and minor gridlines on the first chart in ChartSpace1.

```
Sub EnableGridlines()  
    Dim chConstants  
    Dim axValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to refer to the value axis.  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
  
    ' Enable the major gridlines on the value axis.  
    axValueAxis.HasMajorGridlines = True  
  
    ' Enable the major gridlines on the value axis.  
    axValueAxis.HasMinorGridlines = True  
End Sub
```



# HasMinorGridlines Property

**True** if the specified axis has minor gridlines. Note that any axis can have gridlines. Read/write **Boolean**.

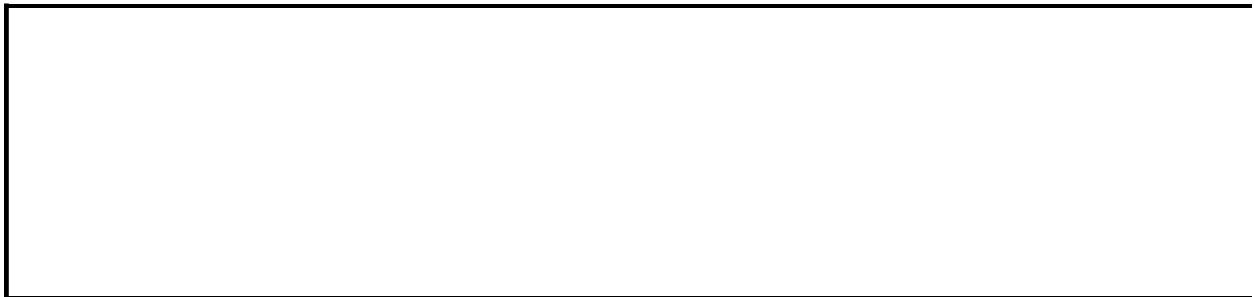
*expression*.**HasMinorGridlines**

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

## Example

This example turns on the major and minor gridlines on the first chart in ChartSpace1.

```
Sub EnableGridlines()  
    Dim chConstants  
    Dim axValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to refer to the value axis.  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
  
    ' Enable the major gridlines on the value axis.  
    axValueAxis.HasMajorGridlines = True  
  
    ' Enable the major gridlines on the value axis.  
    axValueAxis.HasMinorGridlines = True  
End Sub
```



# HasMultipleCharts Property

-

Returns or sets a **Boolean** that indicates whether the specified **ChartSpace** contains multiple charts. The default value is **False**. Read/write.

*expression*.**HasMultipleCharts**

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

---

---



# HasPassiveAlerts Property

-

Returns or sets whether passive alerts are to be used when a non-critical error in encountered. The default value is **True**. Read/write **Boolean**.

*expression*.**HasPassiveAlerts**

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

## Remarks

Passive alerts are presented as a small exclamation symbol in the lower left corner of the chart control. Clicking the symbol displays more information about the non-critical error that has occurred. This feature is similar to the Microsoft Internet Explorer feature of displaying an alert symbol in the status bar when script errors have occurred.

--

# HasPercentage Property

-

**True** if every data label for the specified series or chart currently displays its percentage value. The default value is **False**. This property is available only for pie, doughnut, and stacked charts. Read/write **Boolean**.

*expression*.**HasPercentage**

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

## Remarks

Data label components are always displayed in the following order:  
[SeriesName] [CategoryName] [Value] [BubbleSize] [Percentage].

## Example

This example adds data labels showing percentage value to a series. Note that `Charts(0)` must refer to a pie, doughnut, or stacked chart.

```
Sub AddPercentabelLabels()  
    Dim dl  
  
    Set dl = ChartSpace1.Charts(0).SeriesCollection(0).DataLabelsCol  
  
    ' Display percentage labels.  
    dl.HasPercentage = True  
End Sub
```



# HasPlotDetails Property

**True** if detail fields will be plotted when the chart does not contain a category field. The default value is **False**. Read/write **Boolean**.

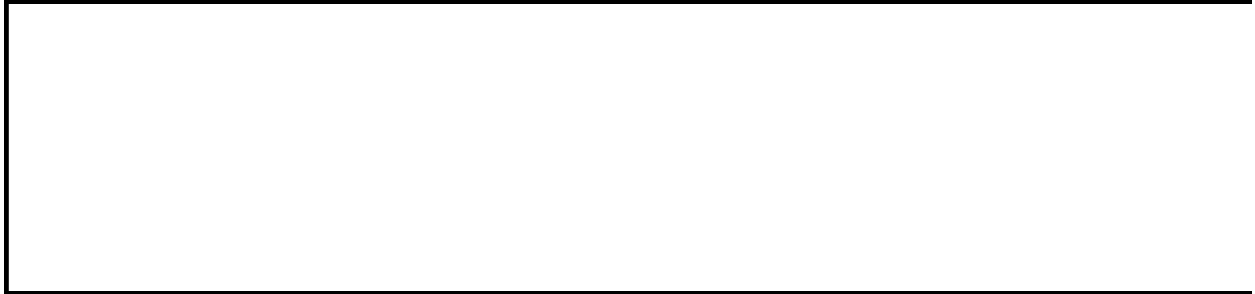
*expression*.**HasPlotDetails**

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

## Example

This example enables detail fields in ChartSpace1 to be plotted when the chart does not contain a category field.

```
ChartSpace1.HasPlotDetails = True
```



# HasRuntimeSelection Property

Returns or sets the selection mode in the charts of the specified chart control. When this property is **True**, the first click selects the inner object; the second click selects the outer object. For example, if this property is **True** and the user clicks the third data point in a data series, only that point is selected. The second time that the user clicks the data point, the entire data series is selected. Setting this property to **False** forces the entire data series to be selected when the user clicks the third data point for the first time. The user has to click the third data point a second time in order to select only that data point. The default value is **True**. Read/write **Boolean**.

*expression*.**HasRuntimeSelection**

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





# HasSelectionMarks Property

-

Set this property to **True** to display selection marks when the user selects an item on a chart. The default value is **False**. Read/write **Boolean**.

*expression*.**HasSelectionMarks**

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

## Remarks

In addition to setting this property to **True**, you must also set the **AllowPropertyToolbox** property to **True** to allow the user to format individual chart elements.

## Example

This example enables the user to select and format individual chart elements in Chartspace1.

```
Sub Allow_Formatting()  
    'Allow the user to display the Commands and Options dialog box.  
    Chartspace1.AllowPropertyToolbox = True  
  
    ' Allow the user to select individual chart elements.  
    Chartspace1.HasSelectionMarks = True  
End Sub
```



# HasSeriesByRows Property

-

Returns or sets how series and categories of the charts in the specified chart control are mapped to the rows and columns of the data source. By default, if the chart is bound to a relational data source, then the series of the chart correspond to the columns of the underlying PivotTable list. The default value is **False**.  
Read/write **Boolean**.

*expression*.**HasSeriesByRows**

*expression* Required. An expression that returns a **ChartSpace** control.



# HasSeriesName Property

**True** if every data label for the specified series or chart currently displays its series name. The default value is **False**. This property is available only for pie, doughnut, and stacked charts. Read/write **Boolean**.

*expression*.**HasSeriesName**

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

## Remarks

Data label components are always displayed in the following order:  
[SeriesName] [CategoryName] [Value] [BubbleSize] [Percentage].

## Example

This example causes the data labels for the specified series to display their category and series names.

```
Sub ShowLabels()  
    Dim dl  
  
    Set dl = ChartSpace1.Charts(0).SeriesCollection(0).DataLabelsCol  
  
    ' Display the category names.  
    dl.HasCategoryName = True  
  
    ' Display the series names.  
    dl.HasSeriesName = True  
End Sub
```



# HasSplit Property

-  
**True** if the scale for the specified axis has a split point between the value of its [SplitMinimum](#) property and the value of its [SplitMaximum](#) property. The default value is **False**. Read/write **Boolean**.

*expression*.**HasSplit**

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



## Example

This example splits the value axis of the first chart in ChartSpace1 and sets the split minimum and split maximum values. The value axis is split, and values between 1000 and 5000 will not be displayed.

```
Sub Split_Value_Axis()  
    Dim chConstants  
    Dim scValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the scaling object of the value axis.  
    Set scValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
  
    ' Add a split to the value axis.  
    scValueAxis.HasSplit = True  
  
    ' Specify the minimum value of the split.  
    scValueAxis.SplitMinimum = 1000  
  
    ' Specify the maximum value for the split.  
    scValueAxis.SplitMaximum = 5000  
End Sub
```



# HasTickLabels Property

**True** if the specified axis has a label at each major tick mark. The default value is **True**. Read/write **Boolean**.

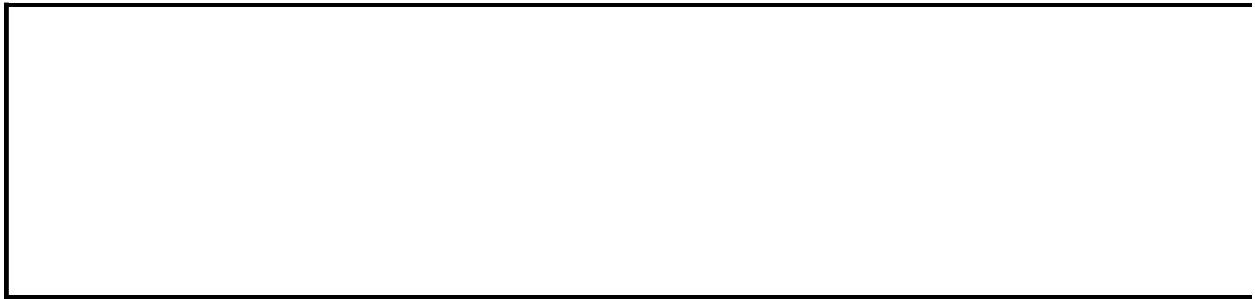
*expression*.**HasTickLabels**

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

## Example

This example turns off tick-mark labels on the category axis on the first chart in ChartSpace1.

```
Sub DisableTickLabels()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Turn off the tick mark labels on the category axis.  
    ChartSpace1.Charts(0).Axes(chConstants.chAxisPositionCategory).H  
End Sub
```



# HasTitle Property

**True** if the specified chart or axis has a title. The default value is **False**.  
Read/write **Boolean**.

*expression*.**HasTitle**

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

## Example

This example causes the title of the specified chart to be displayed and sets the title font.

```
Sub SetChartTitle()  
    ' Enable the chart title for the first chart in ChartSpace1.  
    ChartSpace1.Charts(0).HasTitle = True  
  
    ' Set the title caption.  
    ChartSpace1.Charts(0).Title.Caption = "Monthly Sales"  
  
    ' Set the title font.  
    ChartSpace1.Charts(0).Title.Font.Name = "times new roman"  
End Sub
```



# HasUnifiedScales Property

**True** if all charts in the specified chart control use the same scaling for their axes. The default value is **False**. Read/write **Boolean**.

*expression*.**HasUnifiedScales**

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

## Remarks

This property will return a run-time error when the chart control contains multiple charts with incompatible axes. For example, an error will be returned when the chart control contains an XY (Scatter) chart and a Column chart. This property cannot be set because a Column chart contains a category axis, and the XY (Scatter) chart does not.

When you use the **Add** method to add a new chart to the chart control after setting this property to **True**, the new chart's axes are not automatically unified with the existing charts. You must set this property to **False** and then back to **True** to unify the new chart's axes with the existing charts.

# HasValue Property

**True** if every data label for the specified series or chart currently displays its y-axis value. The default value is **False**. Read/write **Boolean**.

*expression*.**HasValue**

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



## Remarks

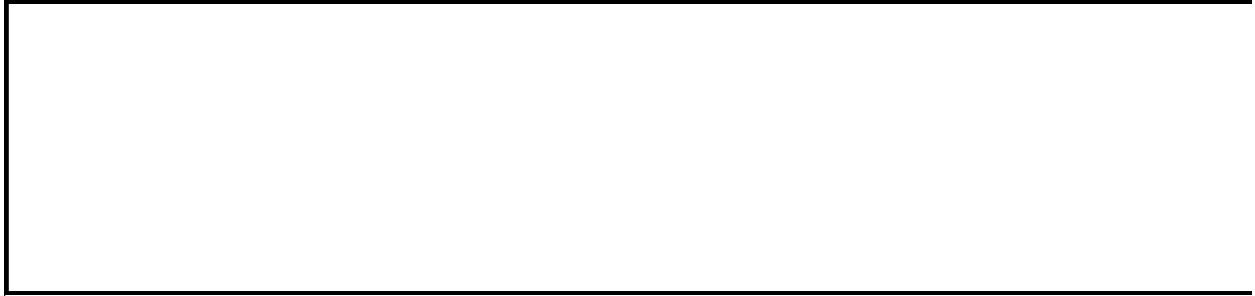
Data label components are always displayed in the following order:  
[SeriesName] [CategoryName] [Value] [BubbleSize] [Percentage].

Setting this property to **True** on a scatter chart or polar chart causes both x-axis values and y-axis values to be displayed.

## Example

This example sets the data labels for the specified series to display their y-axis values.

```
ChartSpace1.Charts(0).SeriesCollection(0).DataLabelsCollection(0).Has
```



# HeaderBackColor Property

-

Returns or sets a **V ariant** indicating the background color of the headings in the data area or an item in the filter area. Read/write.

*expression*.**HeaderBackColor**

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

## Remarks

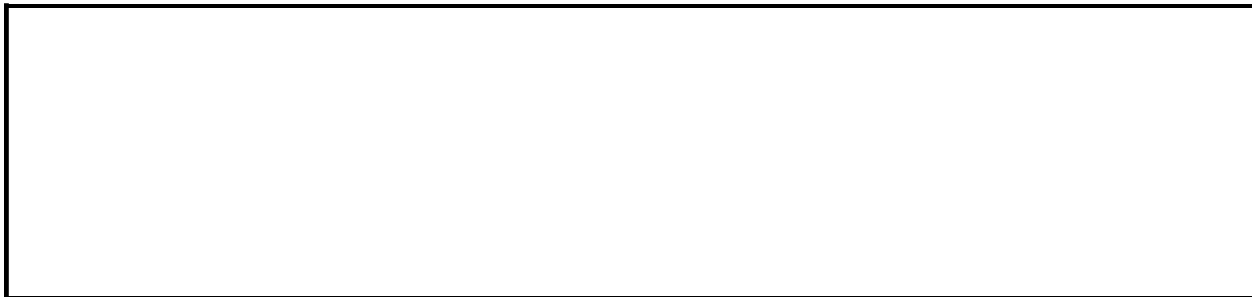
When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

Use the **HeaderFont**, **HeaderForeColor**, and **HeaderHAlignment** properties in addition to this property to format the heading in the data area or an item in the filter area of a PivotTable list.

## Example

This example formats the header for the data area and the filter area of the current view in PivotTable1.

```
Sub SetHeaderProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set the background color.  
    ptView.HeaderBackColor = "Gray"  
  
    ' Set the foreground color.  
    ptView.HeaderForeColor = "Blue"  
  
    ' Set the font.  
    ptView.HeaderFont = "Tahoma"  
  
    ' Set the alignment.  
    ptView.HeaderHAlignment = ptConstants.plHAlignLeft  
  
End Sub
```



# HeaderFont Property

-

Returns a **PivotFont** object representing the font for the headings in the data area or an item in the filter area of the specified view.

*expression*.**HeaderFont**

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

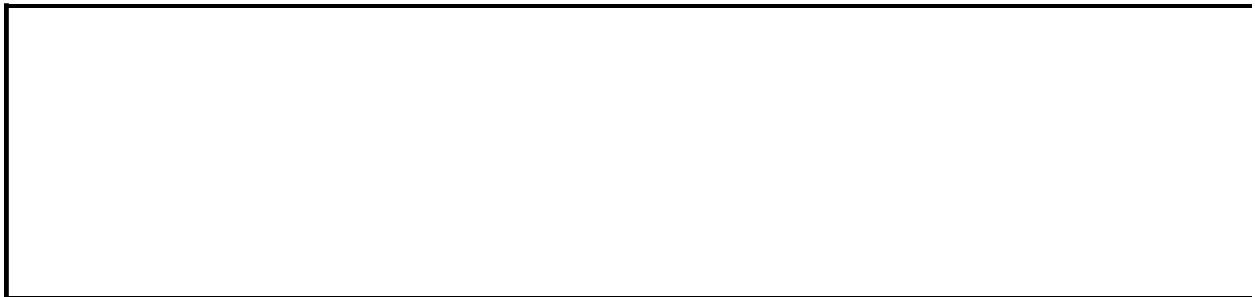
## Remarks

Use the **HeaderBackColor**, **HeaderForeColor**, and **HeaderHAlignment** properties to in addition to this property to format the heading in the data area or an item in the filter area of a PivotTable list.

## Example

This example formats the header for the data area and the filter area of the current view in PivotTable1.

```
Sub SetHeaderProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set the background color.  
    ptView.HeaderBackColor = "Gray"  
  
    ' Set the foreground color.  
    ptView.HeaderForeColor = "Blue"  
  
    ' Set the font.  
    ptView.HeaderFont = "Tahoma"  
  
    ' Set the alignment.  
    ptView.HeaderHAlignment = ptConstants.plHAlignLeft  
  
End Sub
```





# HeaderForeColor Property

-

Returns or sets a **Variant** indicating the foreground color of the headings in the data area or an item in the filter area of the specified view. Read/write.

*expression*.**HeaderForeColor**

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

## Remarks

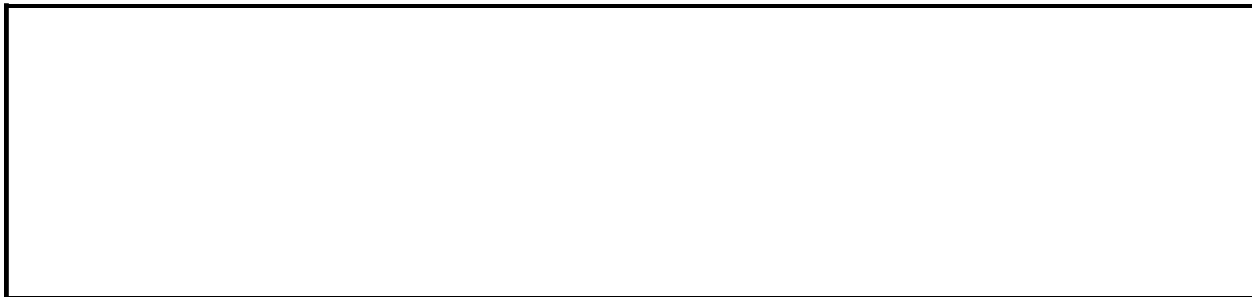
When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

Use the **HeaderBackColor**, **HeaderFont**, and **HeaderHAlignment** properties in addition to this property to format the heading in the data area or an item in the filter area of a PivotTable list.

## Example

This example formats the header for the data area and the filter area of the current view in PivotTable1.

```
Sub SetHeaderProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set the background color.  
    ptView.HeaderBackColor = "Gray"  
  
    ' Set the foreground color.  
    ptView.HeaderForeColor = "Blue"  
  
    ' Set the font.  
    ptView.HeaderFont = "Tahoma"  
  
    ' Set the alignment.  
    ptView.HeaderHAlignment = ptConstants.plHAlignLeft  
  
End Sub
```



▼ [Show All](#)

# HeaderHAlignment Property

Returns or sets a [PivotHAlignmentEnum](#) constant that represents the horizontal alignment of a heading in the data area or an item in the filter area of the specified view. Read/write.

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

*expression*.**HeaderHAlignment**

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

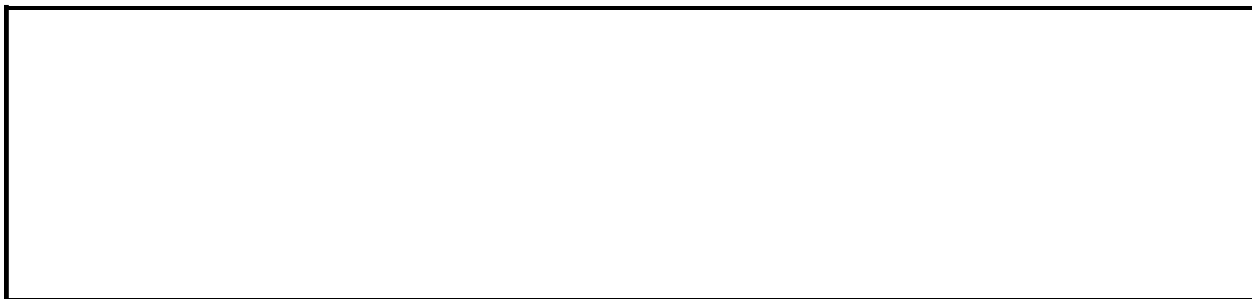
## Remarks

Use the **HeaderBackColor**, **HeaderForeColor**, and **HeaderFont** properties in addition to this property to format the heading in the data area or an item in the filter area of a PivotTable list.

## Example

This example formats the header for the data area and the filter area of the current view in PivotTable1.

```
Sub SetHeaderProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set the background color.  
    ptView.HeaderBackColor = "Gray"  
  
    ' Set the foreground color.  
    ptView.HeaderForeColor = "Blue"  
  
    ' Set the font.  
    ptView.HeaderFont = "Tahoma"  
  
    ' Set the alignment.  
    ptView.HeaderHAlignment = ptConstants.plHAlignLeft  
  
End Sub
```



# HeaderHeight Property

-

Returns a **Long** value that represents the height of the headers for fields on the data axis and the items in the filter field of the specified view. Read-only.

*expression*.**HeaderHeight**

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





# Height Property

Returns or sets the height of the specified object in points.

- Read-only **Double** for the [Window](#) object.
- Read-only **Variant** for the [Range](#) object.
- Read-write **Long** for the [PivotAxisMember](#), [PivotColumnMember](#), [PivotPageMember](#), [PivotRowMember](#), and [PivotTable](#) objects.
- Read-only **Long** for all other objects in the Applies To list.

*expression*.**Height**

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

## Remarks

The **AutoFit** property of the PivotTable list is set to **False** when the value of the **Height** property is changed.

## Example

This example sets the height of the PivotTable list to 36 points.

```
PivotTable1.Object.Height = 36
```



# HeightRatio Property

-

Returns or sets the height ratio for the specified chart in relation to the other charts in the chart workspace. The default value is 100. Read/write **Long**.

*expression*.**HeightRatio**

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

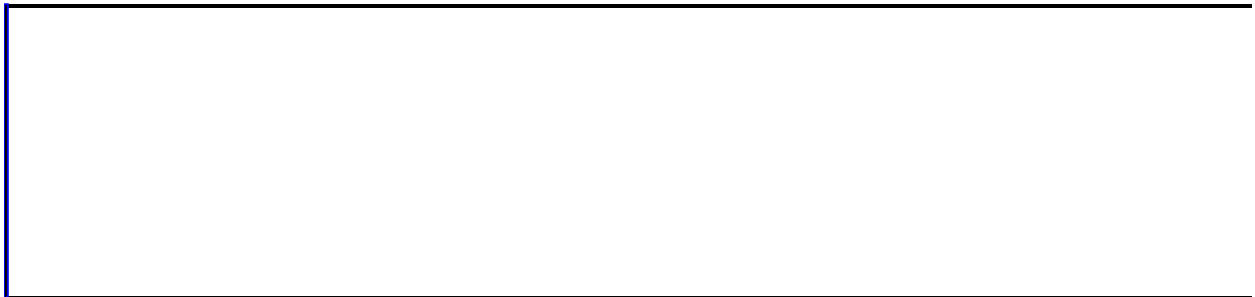
## Remarks

This property sets the height of the specified chart relative to the height of the other charts in the chart workspace. For this property to have any effect, you must have more than one chart in the chart workspace. When more than one chart is displayed, the charts are displayed in a grid (for more information, see the Help topics for the [ChartLayout](#) and [ChartWrapCount](#) properties). Initially, the **HeightRatio** and **WidthRatio** properties are set to 100 for all charts in the grid, and all charts are the same size.

To change the height of charts in the grid, adjust the **HeightRatio** property settings. For example, if each chart is displayed in three rows, all the charts have an initial **HeightRatio** setting of 100. If you want row 3 to be only half the available height, set its **HeightRatio** setting to 200; the remaining half of the chart height is divided between rows 1 and 2. Because the height specified by the **HeightRatio** property is relative, you can set this property for the three rows to 1,1,2; 100,100,200; or 20,20,40, all of which have the same effect.

If the chart workspace contains charts displayed in more than one row, the largest **HeightRatio** setting in each row is used to set the relative height for the entire row.

This property is useful for creating price and volume stock charts in which the volume chart is half the size of the price chart.



# Hidden Property

**True** if the specified range is currently hidden. The range must span an entire column or row. You can use the **EntireRow** and **EntireColumn** properties to return a reference to an entire row or column. Read/write **Boolean**.

*expression*.**Hidden**

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

## Example

This example loops through a row that contains date values. When the month in the cell does not match the current month, the column is hidden. When the month matches the current month and the column is hidden, then the column is unhidden.

```
Sub Hide_Dates()  
    Dim rngLoopRange  
    Dim rngCurrCell  
    Dim ssConstants  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Set range to loop through the range of contiguous  
    ' cells in row 1 starting in column A.  
    Set rngLoopRange = Spreadsheet1.ActiveSheet.Range("A1", _  
        Spreadsheet1.ActiveSheet.Range("A1").End(ssConstants  
  
    ' Loop through the cells.  
    For Each rngCurrCell In rngLoopRange  
  
        ' Hide the column if the month in the current cell does  
        ' not match the current month.  
        If Month(rngCurrCell.Value) <> Month(Date) Then  
            rngCurrCell.EntireColumn.Hidden = True  
  
        ' If the month in the current cell matches the current month  
        ' and the column is hidden, then unhide the column.  
        ElseIf rngCurrCell.EntireColumn.Hidden Then  
            rngCurrCell.EntireColumn.Hidden = False  
        End If  
    Next  
End Sub
```



# HoleSize Property

-

Returns or sets the hole size for the specified doughnut chart. The hole size must be a value from 0 through 90 (the default value is 50). Read/write **Long**.

*expression*.**HoleSize**

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



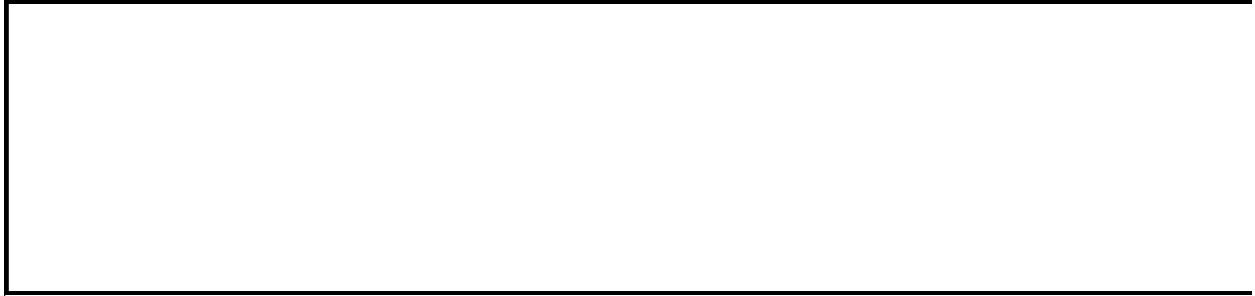
## Remarks

Setting this property to zero (0) changes the doughnut chart to a pie chart (the [Type](#) property for the **ChChart** object returns **chChartTypePieStacked**).

## Example

This example sets hole size for the specified chart. Note that `Charts(0)` must refer to a doughnut chart.

```
ChartSpace1.Charts(0).HoleSize = 25
```



▼ [Show All](#)

# HorizontalAlignment Property

-

Returns or sets a **Variant** representing the horizontal alignment for the specified range. Can be an [XlHAlign](#) constant.

XlHAlign can be one of these XlHAlign constants.

**xlHAlignGeneral**

**xlHAlignCenter**

**xlHAlignLeft**

**xlHAlignRight**

*expression*.**HorizontalAlignment**

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

## Example

This example left-aligns the range A1:A5 on Sheet1.

```
Sub AlignRange()  
    Dim ssConstants  
    Set ssConstants = Spreadsheet1.Constants  
  
    Worksheets("Sheet1").Range("A1:A5").HorizontalAlignment = ssConst  
End Sub
```



# HTMLContainer Property

-  
Returns the HTML **DIV** element that contains the specified section. Read-only.

*expression*.**HTMLContainer**

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



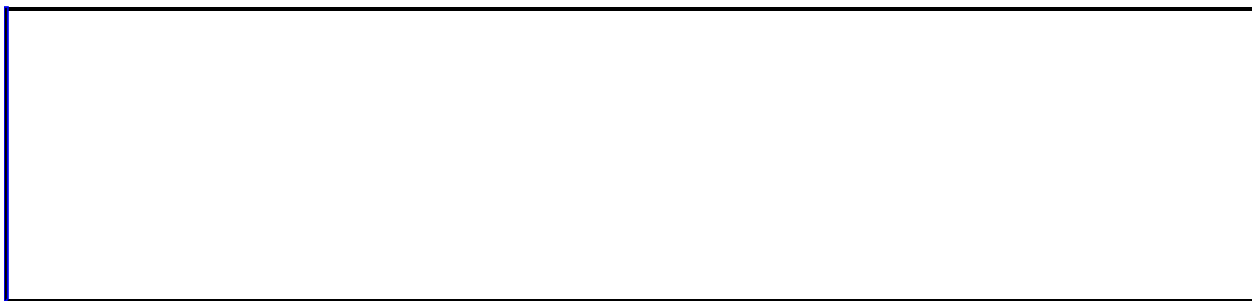
# HTMLData Property

**PivotTable** and **Range** objects: Returns a **String** that represents the specified PivotTable list or range as a properly-formatted HTML string. Read-only.

**Spreadsheet** object: Returns or sets a **String** that represents the spreadsheet data as a properly-formatted HTML string. Read/write.

*expression*.**HTMLData**

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



# HTMLURL Property

-

Returns or sets the URL (Internet address) for the spreadsheet HTML data file.  
Read/write **String**.

*expression*.**HTMLURL**

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





# Hwnd Property

-

Returns a **Long** indicating the top-level window handle of the PivotTable control's window. Read-only.

*expression*.**Hwnd**

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



# Hyperlink Property

-  
**PivotAxisMember**, **PivotColumnMember**, **PivotDetailCell**, **PivotPageMember**, and **PivotRowMember** objects: Returns a **PivotHyperlink** object that represents the hyperlink for the specified object.

**Range** object: Returns a **Hyperlink** object that represents the hyperlink for the specified range.

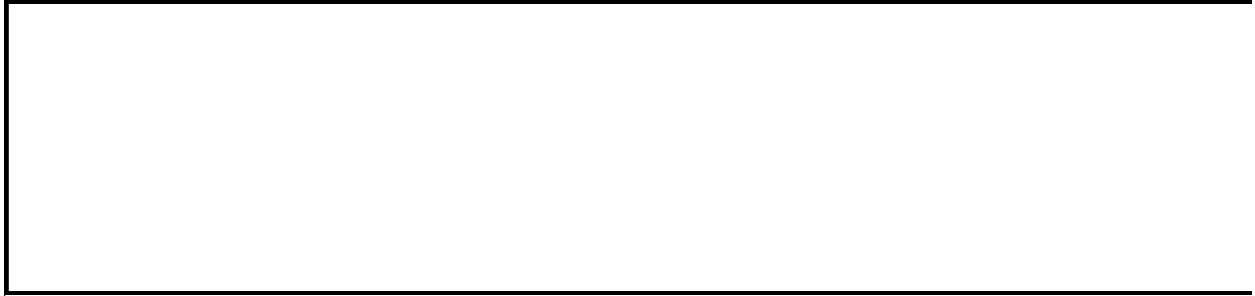
*expression*.**Hyperlink**

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

## Example

This example resolves the hyperlink in cell B15 on the active worksheet, downloads the target document, and then displays the document.

```
Spreadsheet1.ActiveSheet.Range("b15").Hyperlink.Follow
```



# id Property

Returns a **Long** that represents the identifier for a custom drawing object. Read-only.

*expression*.**id**

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



# Inclination Property

-

Returns or sets a **Double** indicating the inclination of the view of the specified three-dimensional (3-D) chart. Valid values range from -90 to 90. Read/write.

*expression*.**Inclination**

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

## Remarks

Setting this property to 90 yields an overhead view of the specified chart.

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then sets the inclination of the view.

```
Sub SetGapDepth()  
    Dim cht3DColumn As ChChart  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Column chart.  
    cht3DColumn.Type = chChartTypeColumn3D  
  
    ' Set the inclination.  
    cht3DColumn.Inclination = 45  
End Sub
```



# Include Property

-

Returns or sets the error bar elements that will be included on the specified chart. The default value is **chErrorBarIncludeBoth**. Read/write [ChartErrorBarIncludeEnum](#).

ChartErrorBarIncludeEnum can be one of these ChartErrorBarIncludeEnum constants.

**chErrorBarIncludeBoth**

**chErrorBarIncludeMinusValues**

**chErrorBarIncludePlusValues**

*expression*.**Include**

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



## Example

This example adds error bars to the specified chart and sets the error bars to display only plus values.

```
Sub Format_ErrorBars()  
    Dim chConstants  
    Dim ebErrorBars  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Add error bars to the first series in the first chart.  
    Set ebErrorBars = ChartSpace1.Charts(0).SeriesCollection(0).Error  
  
    ' Include positive values.  
    ebErrorBars.Include = chConstants.chErrorBarIncludePlusValues  
End Sub
```



# IncludedMembers Property

-

Returns or sets the members to be displayed in the specified field. This property can be set to a single member or a **Variant** array of members. The members can be passed as one or more **PivotMember** objects, member names, or unique member names. Read/write.

*expression*.**IncludedMembers**

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

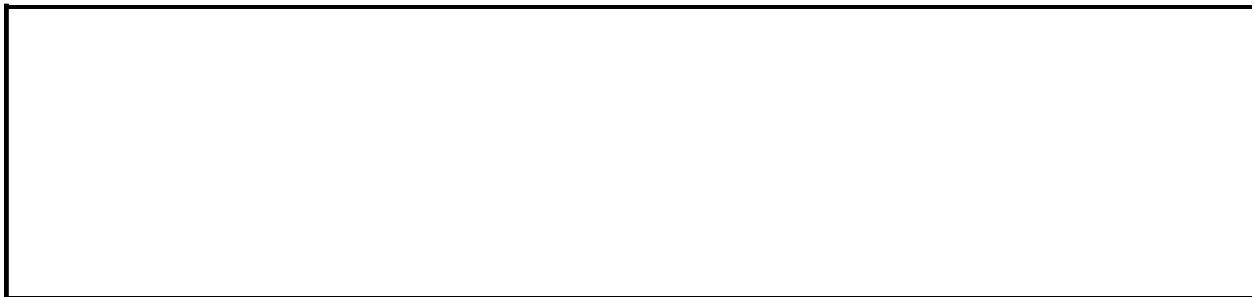
## Remarks

Members not listed when you set this property may still appear in the PivotTable list if their parent member is included. Setting this property clears all previous settings of this property for the specified field. You can set this property to **Empty** (`IncludedMembers = Empty`) or to a zero-length **Variant** array (`IncludedMembers = Array()`) to clear the included members list for the specified field.

## Example

This example sets the included and excluded members of the Store State and Store City fields in PivotTable1.

```
Sub Member_Filtering()  
  
    Dim fldStoreCity  
    Dim fldStoreState  
    Dim ptView  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Store State field.  
    Set fldStoreState = ptView.FieldSets("Store").Fields("Store State")  
  
    ' Set a variable to the Store City field.  
    Set fldStoreCity = ptView.FieldSets("Store").Fields("Store City")  
  
    ' Exclude California and Washington from the Store State field.  
    fldStoreState.ExcludedMembers = Array("CA", "WA")  
  
    ' Include members of the Store City field. Note that the cities  
    ' in states that have been excluded by the previous line. Since  
    ' Store State is a parent to Store City, then the excluded state  
    ' are displayed in the PivotTable.  
    fldStoreCity.IncludedMembers = Array("Los Angeles", "San Diego",  
                                           "Seattle", "Spokane")  
  
End Sub
```



# Index Property

-

Returns the index number of the specified object within the collection of similar objects. Read/write **Long** for the [ChSeries](#) object; read-only **Long** for all other objects in the Applies To list.

*expression*.**Index**

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

## Remarks

You can set the index number of a **ChSeries** object. The specified series is moved to the specified position, and any other series are reordered as necessary.

## Example

This example moves series 1 to the fourth index position of an XY chart containing 6 series. During the move, series 2 through 4 are reordered to series 1 through 3. Series 0 and series 5 stay in the same position.

```
ChartSpace1.Charts(0).SeriesCollection(1).Index = 4
```



▼ [Show All](#)



# Interior Property

▸ [Interior property as it applies to the \*\*ChartSpace\*\*, \*\*ChChart\*\*, \*\*ChChartDraw\*\*, \*\*ChDataLabel\*\*, \*\*ChDataLabels\*\*, \*\*ChLegend\*\*, \*\*ChPlotArea\*\*, \*\*ChPoint\*\*, \*\*ChSegmentBoundary\*\*, \*\*ChSeries\*\*, \*\*ChSurface\*\*, and \*\*ChTitle\*\* objects.](#)

Returns a **ChInterior** object that represents the interior of the specified object.

*expression*.**Interior**

*expression* Required. An expression that returns one of the above objects.

▸ [Interior property as it applies to the \*\*Range\*\* and \*\*TitleBar\*\* objects.](#)

Returns an **Interior** object that represents the interior of the specified object.

*expression*.**Interior**

*expression* Required. An expression that returns one of the above objects.

## Example

This example sets the interior color for the specified series.

```
ChartSpace1.Charts(0).SeriesCollection(0).Interior.Color = "red"
```



# InternalPivotTable Property

-

Returns a **PivotTable** object that is used internally by the chart control to bind to an external data source.

*expression*.**InternalPivotTable**

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



# International Property

Returns information about the current country/region and international settings.  
Read-only **Variant**.

*expression*.**International**(*Index*)

*expression* Required. An expression that returns a **Spreadsheet** or a **ChartSpace** object.

*Index* Required **Long**. The setting to be returned. Can be one of the **XlApplicationInternational** constants listed in the following table.

Index	Type	Meaning
<b>xlCountryCode</b>	<b>Long</b>	Country/region version of the Microsoft Office Web Components.
<b>xlCountrySetting</b>	<b>Long</b>	Current locale setting in Control Panel.
<b>xlDecimalSeparator</b>	<b>String</b>	Decimal separator.
<b>xlThousandsSeparator</b>	<b>String</b>	Zero or thousands separator.
<b>xlListSeparator</b>	<b>String</b>	List separator.
<b>xlUpperCaseRowLetter</b>	<b>String</b>	Uppercase row letter (for R1C1-style references).
<b>xlUpperCaseColumnLetter</b>	<b>String</b>	Uppercase column letter.
<b>xlLowerCaseRowLetter</b>	<b>String</b>	Lowercase row letter.
<b>xlLowerCaseColumnLetter</b>	<b>String</b>	Lowercase column letter.
<b>xlLeftBracket</b>	<b>String</b>	Character used instead of the left bracket (I) in R1C1-style relative references.
<b>xlRightBracket</b>	<b>String</b>	Character used instead of the right bracket (J) in R1C1-style references.
<b>xlLeftBrace</b>	<b>String</b>	Character used instead of the left brace (I) in array literals.
<b>xlRightBrace</b>	<b>String</b>	Character used instead of the right brace (J) in array literals.

		(}) in array literals.
<b>xlColumnSeparator</b>	<b>String</b>	Character used to separate columns in array literals.
<b>xlRowSeparator</b>	<b>String</b>	Character used to separate rows in array literals.
<b>xlAlternateArraySeparator</b>	<b>String</b>	Alternate array item separator to be used if the current array separator is the same as the decimal separator.
<b>xlDateSeparator</b>	<b>String</b>	Date separator (/).
<b>xlTimeSeparator</b>	<b>String</b>	Time separator (:).
<b>xlYearCode</b>	<b>String</b>	Year symbol in number formats (y).
<b>xlMonthCode</b>	<b>String</b>	Month symbol (m).
<b>xlDayCode</b>	<b>String</b>	Day symbol (d).
<b>xlHourCode</b>	<b>String</b>	Hour symbol (h).
<b>xlMinuteCode</b>	<b>String</b>	Minute symbol (m).
<b>xlSecondCode</b>	<b>String</b>	Second symbol (s).
<b>xlCurrencyCode</b>	<b>String</b>	Currency symbol.
<b>xlGeneralFormatName</b>	<b>String</b>	Name of the General number format.
<b>xlCurrencyDigits</b>	<b>Long</b>	Number of decimal digits to be used in currency formats.
		Currency format for negative currency values: 0 = <i>(symbolx)</i> or <i>(xsymbol)</i> 1 = <i>-symbolx</i> or <i>-xsymbol</i> 2 = <i>symbol-x</i> or <i>x-symbol</i> 3 = <i>symbolx-</i> or <i>xsymbol-</i> where <i>symbol</i> is the currency symbol of the country or region. Note that the position of the currency symbol is determined by <b>xlCurrencyBefore</b> .
<b>xlCurrencyNegative</b>	<b>Long</b>	
<b>xlNoncurrencyDigits</b>	<b>Long</b>	Number of decimal digits to be used in non-currency formats.
<b>xlMonthNameChars</b>	<b>Long</b>	Always returns three characters for backward compatibility. Abbreviated month names are read from Microsoft Windows and can be any length.

<b>xlWeekdayNameChars</b>	<b>Long</b>	Always returns three characters for backward compatibility. Abbreviated weekday names are read from Microsoft Windows and can be any length.
<b>xlDateOrder</b>	<b>Long</b>	Order of date elements: 0 = month-day-year 1 = day-month-year 2 = year-month-day
<b>xl24HourClock</b>	<b>Boolean</b>	<b>True</b> if you're using 24-hour time; <b>False</b> if you're using 12-hour time.
<b>xlNonEnglishFunctions</b>	<b>Boolean</b>	<b>True</b> if you're not displaying functions in English.
<b>xlMetric</b>	<b>Boolean</b>	<b>True</b> if you're using the metric system; <b>False</b> if you're using the English measurement system.
<b>xlCurrencySpaceBefore</b>	<b>Boolean</b>	<b>True</b> if a space is added before the currency symbol.
<b>xlCurrencyBefore</b>	<b>Boolean</b>	<b>True</b> if the currency symbol precedes the currency values; <b>False</b> if it follows them.
<b>xlCurrencyMinusSign</b>	<b>Boolean</b>	<b>True</b> if you're using a minus sign for negative numbers; <b>False</b> if you're using parentheses.
<b>xlCurrencyTrailingZeros</b>	<b>Boolean</b>	<b>True</b> if trailing zeros are displayed for zero currency values.
<b>xlCurrencyLeadingZeros</b>	<b>Boolean</b>	<b>True</b> if leading zeros are displayed for zero currency values.
<b>xlMonthLeadingZero</b>	<b>Boolean</b>	<b>True</b> if a leading zero is displayed in months (when months are displayed as numbers).
<b>xlDayLeadingZero</b>	<b>Boolean</b>	<b>True</b> if a leading zero is displayed in days.
<b>xl4DigitYears</b>	<b>Boolean</b>	<b>True</b> if you're using four-digit years; <b>False</b> if you're using two-digit years.
<b>xlMDY</b>	<b>Boolean</b>	<b>True</b> if the date order is month-day-year for dates displayed in the long form; <b>False</b> if the date order is day-month-year.

**xlTimeLeadingZero**

**Boolean** **True** if a leading zero is displayed in times.

## Remarks

Symbols, separators, and currency formats shown in the preceding table may differ from those used in your language or geographic location and may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--



# IsConsistent Property

-  
Returns a **Boolean** that indicates whether the source data set is consistent with the specified **PivotData** object. Read-only.

*expression*.**IsConsistent**

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

## Remarks

This property returns **False** when changes to the data set have been committed, but the totals have not been updated. When this occurs, you can use the **Refresh** method of the **PivotTable** object to update the totals.

# IsDataBound Property

-

Returns or sets a **Boolean** that determines whether the specified worksheet is bound to an outside data source. Read/write.

*expression*.**IsDataBound**

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



# IsDirty Property

-

Returns or sets a **Boolean** that indicates whether the PivotTable list has changed since the last time it was saved. Read/write.

*expression*.**IsDirty**

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



# IsDisplayingEquation Property

**True** if the equation for the trendline for the specified series is displayed on the chart (in the same data label as the R-squared value). The default value is **True**.  
Read/write **Boolean**.

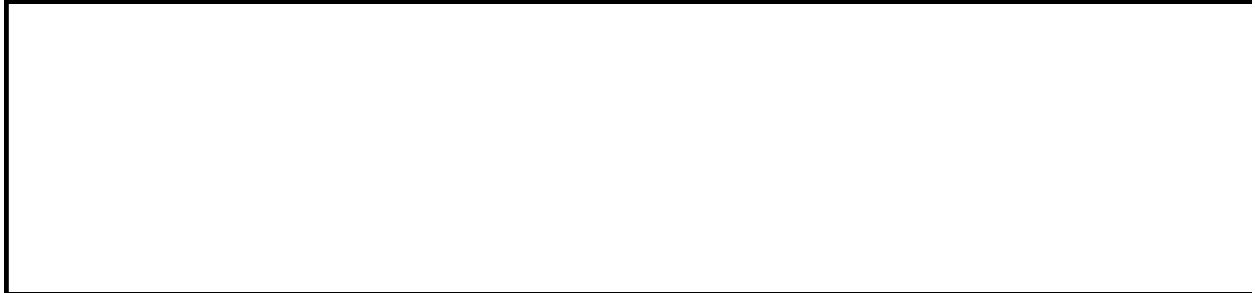
*expression*.**IsDisplayingEquation**

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

## Example

This example adds a trendline to the specified series. Only the trendline equation is displayed.

```
Sub AddTrendLine()  
    Dim trndline  
  
    ' Add a trendline to the first series in the first chart in Char  
    Set trndline = ChartSpace1.Charts(0).SeriesCollection(0).Trendli  
  
    ' Set the font of the trendline to bold.  
    trndline.DataLabel.Font.Bold = True  
  
    ' Do not display the R-Squared value with the trendline.  
    trndline.IsDisplayingRSquared = False  
  
    ' Display the equation for the trendline.  
    trndline.IsDisplayingEquation = True  
End Sub
```



# IsDisplayingRSquared Property

-

**True** if the R-squared value for the trendline for the specified series is displayed on the chart (in the same data label as the equation). Setting this property to **True** automatically turns on data labels. The default value is **True**. Read/write **Boolean**.

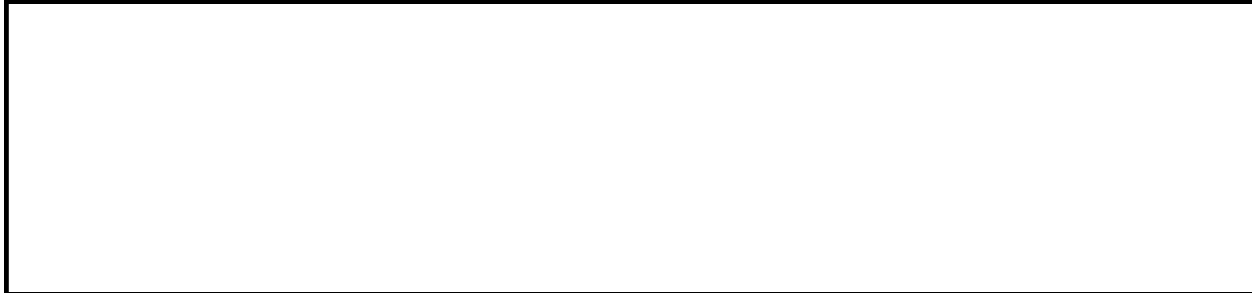
*expression*.**IsDisplayingRSquared**

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

## Example

This example adds a trendline to the specified series. Only the trendline equation is displayed.

```
Sub AddTrendLine()  
    Dim trndline  
  
    ' Add a trendline to the first series in the first chart in Char  
    Set trndline = ChartSpace1.Charts(0).SeriesCollection(0).Trendli  
  
    ' Set the font of the trendline to bold.  
    trndline.DataLabel.Font.Bold = True  
  
    ' Do not display the R-Squared value with the trendline.  
    trndline.IsDisplayingRSquared = False  
  
    ' Display the equation for the trendline.  
    trndline.IsDisplayingEquation = True  
End Sub
```





# IsExpanded Property

-

**True** if the specified section is expanded. Read-only **Boolean**.



*expression*.**IsExpanded**

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

# IsFiltered Property

-  
**True** if the current filter is applied. The default value is **True**. Read/write **Boolean**.

*expression*.**IsFiltered**

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

# IsFilterOn Property

-  
**True** if the data page filter is applied. Read/write **Boolean**.

---

*expression*.**IsFilterOn**

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

▼ [Show All](#)

# IsHyperlink Property

▸ [IsHyperlink property as it applies to the \*\*PivotField\*\* object.](#)

Returns or sets whether the items in the specified field are to be treated as hyperlinks. Set this property to **True** in order to treat the items in the specified field as hyperlinks. The default value is **False**. Read/write **Boolean**.

*expression*.**IsHyperlink**

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

▸ [IsHyperlink property as it applies to the \*\*SchemaField\*\* and \*\*SchemaParameter\*\* objects.](#)

Returns whether or not the specified field contains hyperlinks. Read-only **Boolean**.

*expression*.**IsHyperlink**

*expression* Required. An expression that returns one of the above objects.

## Example

► [As it applies to the PivotField object.](#)

This example converts the items in the Merchant field in PivotTable1 to hyperlinks.

```
PivotTable1.ActiveView.FieldSets("Merchant") _  
    .Fields(0).IsHyperlink = True
```



# IsIncluded Property

**True** if the field is included and active in the specified field set. Read/write **Boolean**.

*expression*.**IsIncluded**

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

## Remarks

Inactive fields are not visible in the PivotTable list. A field set must have at least one field for which the **IsIncluded** property is set to **True**. Attempting to set this property to **False** for the only field where it is set to **True** generates a run-time error. This property is ignored when the field set is used on the filter axis. If the user hides the last field in the user interface, the field set is removed from the axis.



# IsTotal Property

**True** if the specified member is used to display a subtotal. Read-only **Boolean**.

*expression*.**IsTotal**

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

--

# IsValid Property

-

Returns a **Boolean** that indicated whether the specified member is a valid member within the specified context. Read-only.

*expression*.**IsValid**

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

## Example

This example attempts to find a specific warehouse in the Warehouse field set. The user is alerted if the specified warehouse is not found.

```
Sub FindWarehouse()  
  
    Dim ptView  
    Dim ptConstants  
    Dim fsWarehouse  
    Dim pmFound  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Warehouse field set.  
    Set fsWarehouse = ptView.FieldSets("Warehouse")  
  
    ' Set a variable to the results of the FindMember property.  
    Set pmFound = fsWarehouse.FindMember("Quality Distribution, Inc.")  
  
    ' Check to see if the member was found.  
    If pmFound.IsValid = False Then  
  
        ' Alert the user if the member was not found.  
        MsgBox "The specified member does not exist."  
  
    End If  
  
End Sub
```



# Italic Property

-  
**True** if the font style is italic. Read/write **Boolean** for the [ChFont](#) and [PivotFont](#) objects; read/write **Variant** for the [Font](#) object (returns **Null** if some portions of the text are italic and some are not). Use the **IsNull** function to determine whether the return value is **Null**.

*expression*.**Italic**

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

## Example

This example sets the font to italic for the contents of column B.

```
Spreadsheet1.Columns(2).Font.Italic = True
```



▼ [Show All](#)

# Item Property

## ► [Item property as it applies to the \*\*Borders\*\* object.](#)

Returns a **Border** object that represents the specified border.

*expression*.**Item**(*Index*)

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

*Index* Required [XlBordersIndex](#). The border that you want to return.

XlBordersIndex can be one of these XlBordersIndex constants.

**xlEdgeBottom**

**xlEdgeLeft**

**xlEdgeRight**

**xlEdgeTop**

**xlInsideHorizontal**

**xlInsideVertical**

## ► [Item property as it applies to the \*\*ChCategoryLabels\*\* object.](#)

Returns a **ChCategoryLabel** object that represents the specified category label.

*expression*.**Item**(*Level*, *Index*)

*expression* Required. An expression that returns one of the above objects.

*Level* Optional **Long**. The level of category labels that you want to index.

*Index* Required **Variant**. The name or index number of the category label that you want to return.

► [Item property as it applies to the \*\*PivotMemberProperties\*\* and \*\*PivotResultMemberProperties\*\* objects.](#)

Returns a **PivotMemberProperty** object that represents the specified member property.

*expression*.**Item**(*varIndex*)

*expression* Required. An expression that returns one of the above objects.

**varIndex** Required **Variant**. The name or index number of the member property that you want to return.

► [Item property as it applies to the \*\*Range\*\* object.](#)

Returns a **Range** object that represents a specific cell within the specified range of cells.

*expression*.**Item**(*RowIndex*, *ColumnIndex*)

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

**RowIndex** Required **Variant**. The row index of the cell that you want to return.

**ColumnIndex** Optional **Variant**. The column index of the cell that you want to return.

► [Item property as it applies to all other objects in the \*\*Applies To\*\* list.](#)

Returns a single object from the specified collection. The **Item** property is the default property for most collections, so it is usually not necessary to explicitly use this property. For more information about returning a single member of a collection, see [Returning an Object from a Collection](#).

*expression*.**Item**(*Index*)



*expression* An expression that returns an object in the Applies To list.

***Index*** Required **Long** or **Variant**, depending on the object specified in *expression* (for a list of allowed argument types by object, see the Object Browser). The name or index number of the object that will be returned.

--

# ItemCount Property

-

Returns a **Long** specifying the number of category labels for the specified chart axis. Read-only.

*expression*.**ItemCount**(*Level*)

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

**Level** Optional **Long**. The level of category labels to return the count of. By default, the count of the innermost level of labels is returned.



# JoinType Property

-  
Returns or sets the way a query will be formed. Read/write [DscJoinTypeEnum](#).

DscJoinTypeEnum can be one of these DscJoinTypeEnum constants.

**dscInnerJoin**

**dscLeftOuterJoin**

**dscRightOuterJoin**

*expression*.**JoinType**

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

## Remarks

You can use this property only if the [PageRelationship](#) object's [Type](#) property is set to **dscLookup**.

▼ [Show All](#)

# Label Property

▸ [Label property as it applies to the \*\*PivotAxis\*\*, \*\*PivotDataAxis\*\*, \*\*PivotFilterAxis\*\*, \*\*PivotGroupAxis\*\*, and \*\*PivotView\*\* objects.](#)

Returns a **PivotLabel** object that represents the view label or the axis label.

*expression*.**Label**

*expression* Required. An expression that returns one of the above objects.

▸ [Label property as it applies to the \*\*PivotData\*\*, \*\*PivotResultAxis\*\*, \*\*PivotResultColumnAxis\*\*, \*\*PivotResultDataAxis\*\*, \*\*PivotResultFilterAxis\*\*, \*\*PivotResultGroupAxis\*\*, \*\*PivotResultPageAxis\*\*, and \*\*PivotResultRowAxis\*\* objects.](#)

Returns a **PivotResultLabel** object that represents the axis label.

*expression*.**Label**

*expression* Required. An expression that returns one of the above objects.



▼ [Show All](#)

# LanguageID Property

Returns a **Long** representing the locale identifier (LCID) for the install language, the user interface language, or the Help language. Read-only.

*expression*.**LanguageID**(*id*)

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

*id* Required [MsoAppLanguageID](#).

MsoAppLanguageID can be one of these MsoAppLanguageID constants.

The language mode that the component is using. This setting affects languages that can be displayed and edited, available language-specific features, number styles, currency settings, and so forth.

If none of the supported languages are used, the host application will not be configured to support right-to-left and East Asian languages. The supported languages are as follows:

Arabic

Farsi

Hebrew

Japanese

Korean

Simplified Chinese

**msoLanguageIDExeMode**



Traditional Chinese

Urdu

Yiddish

**msoLanguageIDHelp**

The language used for online Help.

**msoLanguageIDInstall**

The language settings used by the Microsoft Office Web Components to set up defaults.

**msoLanguageIDUI**

The language used by the host application's user interface.

**msoLanguageIDUIPrevious**

The language setting for the user interface when a given computer was last rebooted. A program or add-in can use this to determine whether the user interface language has changed.



▼ [Show All](#)

# LanguagePreferredForEditing Property

Returns **True** if the value for the **msoLanguageID** constant has been identified in the registry as a preferred language for editing. Read-only **Boolean**.

*expression*.**LanguagePreferredForEditing**(*lid*)

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

*lid* Required [MsoLanguageID](#). The language to check for in the registry.

MsoLanguageID can be one of these MsoLanguageID constants.

**msoLanguageIDAfrikaans**

**msoLanguageIDAlbanian**

**msoLanguageIDAmharic**

**msoLanguageIDArabic**

**msoLanguageIDArabicAlgeria**

**msoLanguageIDArabicBahrain**

**msoLanguageIDArabicEgypt**

**msoLanguageIDArabicIraq**

**msoLanguageIDArabicJordan**

**msoLanguageIDArabicKuwait**

**msoLanguageIDArabicLebanon**

**msoLanguageIDArabicLibya**

**msoLanguageIDArabicMorocco**

**msoLanguageIDArabicOman**

**msoLanguageIDArabicQatar**

**msoLanguageIDArabicSyria**

**msoLanguageIDArabicTunisia**

**msoLanguageIDArabicUAE**  
**msoLanguageIDArabicYemen**  
**msoLanguageIDArmenian**  
**msoLanguageIDAssamese**  
**msoLanguageIDAzeriCyrillic**  
**msoLanguageIDAzeriLatin**  
**msoLanguageIDBasque**  
**msoLanguageIDBelgianDutch**  
**msoLanguageIDBelgianFrench**  
**msoLanguageIDBengali**  
**msoLanguageIDBrazilianPortuguese**  
**msoLanguageIDBulgarian**  
**msoLanguageIDBurmese**  
**msoLanguageIDByelorussian**  
**msoLanguageIDCatalan**  
**msoLanguageIDCherokee**  
**msoLanguageIDChineseHongKong**  
**msoLanguageIDChineseMacao**  
**msoLanguageIDChineseSingapore**  
**msoLanguageIDCroatian**  
**msoLanguageIDCzech**  
**msoLanguageIDDanish**  
**msoLanguageIDDutch**  
**msoLanguageIDEnglishAUS**  
**msoLanguageIDEnglishBelize**  
**msoLanguageIDEnglishCanadian**  
**msoLanguageIDEnglishCaribbean**  
**msoLanguageIDEnglishIreland**  
**msoLanguageIDEnglishJamaica**  
**msoLanguageIDEnglishNewZealand**  
**msoLanguageIDEnglishPhilippines**  
**msoLanguageIDEnglishSouthAfrica**  
**msoLanguageIDEnglishTrinidad**

**msoLanguageIDEnglishUK**  
**msoLanguageIDEnglishUS**  
**msoLanguageIDEnglishZimbabwe**  
**msoLanguageIDEstonian**  
**msoLanguageIDFaeroese**  
**msoLanguageIDFarsi**  
**msoLanguageIDFinnish**  
**msoLanguageIDFrench**  
**msoLanguageIDFrenchCameroon**  
**msoLanguageIDFrenchCanadian**  
**msoLanguageIDFrenchCotedIvoire**  
**msoLanguageIDFrenchLuxembourg**  
**msoLanguageIDFrenchMali**  
**msoLanguageIDFrenchMonaco**  
**msoLanguageIDFrenchReunion**  
**msoLanguageIDFrenchSenegal**  
**msoLanguageIDFrenchWestIndies**  
**msoLanguageIDFrenchZaire**  
**msoLanguageIDFrisianNetherlands**  
**msoLanguageIDGaelicIreland**  
**msoLanguageIDGaelicScotland**  
**msoLanguageIDGalician**  
**msoLanguageIDGeorgian**  
**msoLanguageIDGerman**  
**msoLanguageIDGermanAustria**  
**msoLanguageIDGermanLiechtenstein**  
**msoLanguageIDGermanLuxembourg**  
**msoLanguageIDGreek**  
**msoLanguageIDGujarati**  
**msoLanguageIDHebrew**  
**msoLanguageIDHindi**  
**msoLanguageIDHungarian**  
**msoLanguageIDIcelandic**

**msoLanguageIDIndonesian**  
**msoLanguageIDInuktitut**  
**msoLanguageIDItalian**  
**msoLanguageIDJapanese**  
**msoLanguageIDKannada**  
**msoLanguageIDKashmiri**  
**msoLanguageIDKazakh**  
**msoLanguageIDKhmer**  
**msoLanguageIDKirghiz**  
**msoLanguageIDKonkani**  
**msoLanguageIDKorean**  
**msoLanguageIDLao**  
**msoLanguageIDLatvian**  
**msoLanguageIDLithuanian**  
**msoLanguageIDMacedonian**  
**msoLanguageIDMalayalam**  
**msoLanguageIDMalayBruneiDarussalam**  
**msoLanguageIDMalaysian**  
**msoLanguageIDMaltese**  
**msoLanguageIDManipuri**  
**msoLanguageIDMarathi**  
**msoLanguageIDMexicanSpanish**  
**msoLanguageIDMixed**  
**msoLanguageIDMongolian**  
**msoLanguageIDNepali**  
**msoLanguageIDNone**  
**msoLanguageIDNoProofing**  
**msoLanguageIDNorwegianBokmol**  
**msoLanguageIDNorwegianNynorsk**  
**msoLanguageIDOriya**  
**msoLanguageIDOromo**  
**msoLanguageIDPolish**  
**msoLanguageIDPortuguese**

msoLanguageIDPunjabi  
msoLanguageIDRhaetoRomanic  
msoLanguageIDRomanian  
msoLanguageIDRomanianMoldova  
msoLanguageIDRussian  
msoLanguageIDRussianMoldova  
msoLanguageIDSamiLappish  
msoLanguageIDSanskrit  
msoLanguageIDSerbianCyrillic  
msoLanguageIDSerbianLatin  
msoLanguageIDSesotho  
msoLanguageIDSimplifiedChinese  
msoLanguageIDSindhi  
msoLanguageIDSlovak  
msoLanguageIDSlovenian  
msoLanguageIDSorbian  
msoLanguageIDSpanish  
msoLanguageIDSpanishArgentina  
msoLanguageIDSpanishBolivia  
msoLanguageIDSpanishChile  
msoLanguageIDSpanishColombia  
msoLanguageIDSpanishCostaRica  
msoLanguageIDSpanishDominicanRepublic  
msoLanguageIDSpanishEcuador  
msoLanguageIDSpanishElSalvador  
msoLanguageIDSpanishGuatemala  
msoLanguageIDSpanishHonduras  
msoLanguageIDSpanishModernSort  
msoLanguageIDSpanishNicaragua  
msoLanguageIDSpanishPanama  
msoLanguageIDSpanishParaguay  
msoLanguageIDSpanishPeru  
msoLanguageIDSpanishPuertoRico

msoLanguageIDSpanishUruguay  
msoLanguageIDSpanishVenezuela  
msoLanguageIDSutu  
msoLanguageIDSwahili  
msoLanguageIDSwedish  
msoLanguageIDSwedishFinland  
msoLanguageIDSwissFrench  
msoLanguageIDSwissGerman  
msoLanguageIDSwissItalian  
msoLanguageIDTajik  
msoLanguageIDTamil  
msoLanguageIDTatar  
msoLanguageIDTelugu  
msoLanguageIDThai  
msoLanguageIDTibetan  
msoLanguageIDTigrignaEritrea  
msoLanguageIDTigrignaEthiopic  
msoLanguageIDTraditionalChinese  
msoLanguageIDTsonga  
msoLanguageIDTswana  
msoLanguageIDTurkish  
msoLanguageIDTurkmen  
msoLanguageIDUkrainian  
msoLanguageIDUrdu  
msoLanguageIDUzbekCyrillic  
msoLanguageIDUzbekLatin  
msoLanguageIDVenda  
msoLanguageIDVietnamese  
msoLanguageIDWelsh  
msoLanguageIDXhosa  
msoLanguageIDZulu



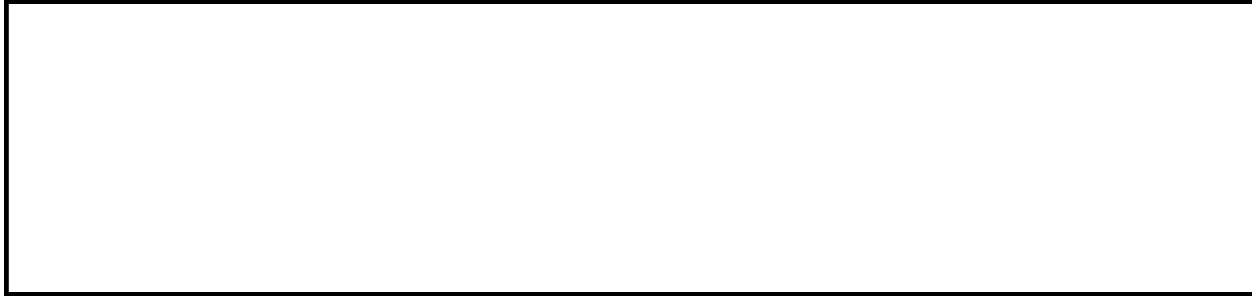
## Remarks

You must test all valid **msoLanguageID** values to enumerate the set of preferred languages.

## Example

The following example tests whether the U.S. English language is registered as a preferred editing language.

```
If Spreadsheet1.LanguageSettings. _  
    LanguagePreferredForEditing(msoLanguageIDEnglishUS) Then  
    MsgBox "U.S. English is one of the chosen editing languages."  
End If
```



# LanguageSettings Property

-

Returns a [OWCLanguageSettings](#) object, which contains information about the language settings in the specified **Spreadsheet** or **Chart** Workspace control.

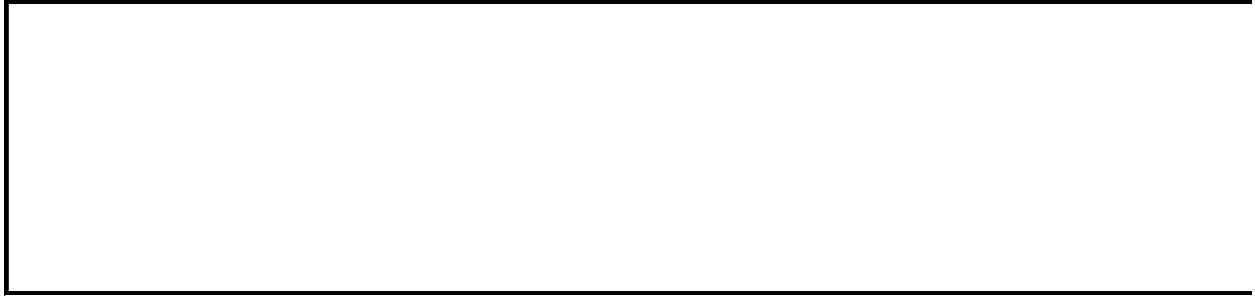
*expression*.**LanguageSettings**

*expression* Required. An expression that returns a **ChartSpace** or a **Spreadsheet** object.

## Example

The following example displays the locale identifier (LCID) for the current language used for the user interface by Spreadsheet1.

```
MsgBox Spreadsheet1.LanguageSettings.LanguageID(msoLanguageIDUI)
```



▼ [Show All](#)

# Left Property

▸ [Left property as it applies to the \*\*ChartSpace\*\*, \*\*ChCategoryLabel\*\*, \*\*ChChartField\*\*, \*\*ChDataLabel\*\*, \*\*ChDataLabels\*\*, \*\*ChErrorBars\*\*, \*\*ChLegendEntry\*\*, \*\*ChPoint\*\*, \*\*ChSeries\*\*, \*\*ChTrendline\*\*, \*\*PivotAggregate\*\*, \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotDetailCell\*\*, \*\*PivotPageMember\*\*, \*\*PivotResultAxis\*\*, \*\*PivotResultColumnAxis\*\*, \*\*PivotResultDataAxis\*\*, \*\*PivotResultFilterAxis\*\*, \*\*PivotResultGroupAxis\*\*, \*\*PivotResultLabel\*\*, \*\*PivotResultPageAxis\*\*, \*\*PivotResultRowAxis\*\*, \*\*PivotRowMember\*\*, and \*\*PivotTable\*\* objects.](#)

Returns (or sets, depending on the object) a **Long** that represents the left edge of the specified object. Read/write.

*expression*.**Left**

*expression* Required. An expression that returns one of the above objects.

▸ [Left property as it applies to the \*\*PivotData\*\* object.](#)

Returns or sets a **PivotColumnMember** object that represents the member immediately above the leftmost visible column. Read/write.

*expression*.**Left**

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

▸ [Left property as it applies to the \*\*Range\*\* object.](#)

Returns a **Variant** that represents the distance from the left edge of the spreadsheet to the left edge of the specified range. This value can be negative if the range is outside the visible range. Read-only.

*expression*.**Left**

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

► [Left property as it applies to the \*\*Window\*\* object.](#)

Returns a **Double** that represents the left edge of the specified window. Read-only.

*expression*.**Left**

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

## Example

This example scrolls to the left through the spreadsheet window to display cell F1 if this cell is currently to the left of the visible range.

```
Sub ScrollSheet
    Dim rngScroll
    Set rngScroll = Spreadsheet1.Range("f1")
    If rngScroll.Left < 0 Then Spreadsheet1.ActiveSheet.Scroll rngScr
End Sub
```





# Left2 Property

-

Returns a **Long** value that represents the distance from the left side of the PivotTable list to the data area. This property always returns 1. Read-only.

*expression*.**Left2**

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



# LeftOffset Property

-

Returns or sets a **Long** value that represents the number of pixels to scroll the data area to the left. Read/write.

*expression*.**LeftOffset**

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

## Remarks

Use the **TopOffset** to scroll the data area down.

## Example

This example scrolls the data area of PivotTable1 down 45 pixels and left 45 pixels.

```
Sub ScrollDataArea()  
    Dim ptData  
    Set ptData = PivotTable1.ActiveData  
    ' Scroll 45 pixels down.  
    ptData.TopOffset = 45  
    ' Scroll the data area to the left.  
    ptData.LeftOffset = 45  
End Sub
```



# Legend Property

-

Returns a [ChLegend](#) object that represents the legend for the specified chart.

*expression*.**Legend**

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

## Example

This example causes the specified chart to display its legend and then sets the legend font.

```
Sub AddLegend()  
    ' Enable the legend for the first chart in Chartspace1.  
    ChartSpace1.Charts(0).HasLegend = True  
  
    ' Set the font for the legend.  
    ChartSpace1.Charts(0).Legend.Font.Name = "Times New Roman"  
End Sub
```



# LegendEntries Property

-  
Returns a [ChLegendEntries](#) collection for the specified legend.

*expression*.**LegendEntries**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).



## Example

This example hides the specified legend entry.

```
ChartSpace1.Charts(0).Legend.LegendEntries(1).Visible = False
```



# Length Property

-  
Returns or sets the maximum field or parameter length. Read/write **Long**.

*expression*.**Length**

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

--

# Level Property

-

Returns a **Long** specifying the level that the specified category label appears on in a hierarchical category axis. The outermost level is level zero (0). Read-only.

*expression*.**Level**

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

---

---

# LevelCount Property

-

Returns a **Long** indicating the number of levels in a hierarchical chart axis.  
Read-only.

*expression*.**LevelCount**

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



# LightNormal Property

-

Returns or sets a **Double** specifying the amount that the light is bent from 90 degrees in a three-dimensional (3-D) chart. Valid values range from 0 to 1. Read/write.

*expression*.**LightNormal**

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

## Remarks

Setting this property to 0 results in a flat look for your chart, while setting this property to 0.5 yields a more three-dimensional look.

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then sets the **LightNormal** property of the chart.

```
Sub SetGapDepth()  
    Dim cht3DColumn As ChChart  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Column chart.  
    cht3DColumn.Type = chChartTypeColumn3D  
  
    cht3DColumn.LightNormal = 0.8  
End Sub
```



# Line Property

-  
Returns a [ChLine](#) object that you can use to change the appearance of the specified chart element.

*expression*.**Line**

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



## Remarks

When used with other properties, the **Line** property can format axes, trendlines, lines on line or stock charts, error bars, and gridlines.

## Example

This example sets the line color for the specified series.

```
ChartSpace1.Charts(0).SeriesCollection(0).Line.Color = "red"
```



# LineStyle Property

-

Returns or sets the border line style for the specified range. Can be one of the **LineStyleEnum** constants; returns **Null** if the borders do not all have the same style. Use the **IsNull** function to determine whether the return value is **Null**. Read/write **Variant**.

*expression*.**LineStyle**

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

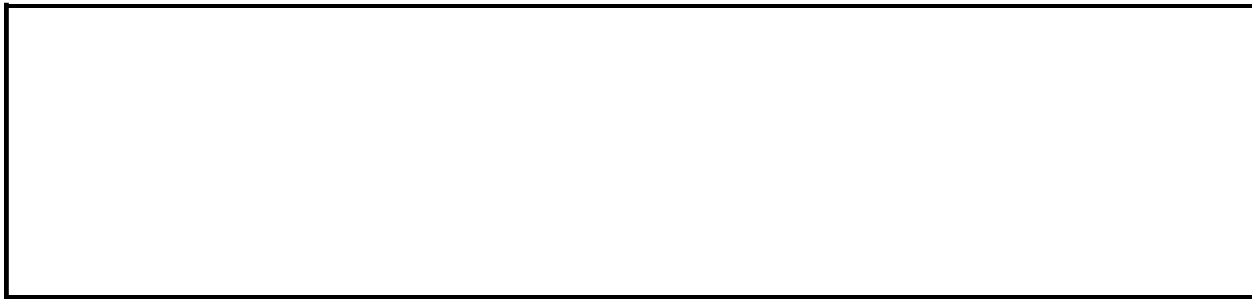
## Remarks

You cannot set this property for a **ChBorder** object that represents a chart element.

## Example

This example puts a green dash-dot border around each cell in the range A1:E5.

```
Sub SetBorder()  
    Dim rngCurrent  
    Dim ssConstants  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    Set rngCurrent = Spreadsheet1.Range("a1:e5")  
  
    rngCurrent.Borders.LineStyle = ssConstants.owcLineStyleDashDot  
    rngCurrent.Borders.Color = "Green"  
End Sub
```



# ListBoundField Property

-

Returns or sets the field that is bound in the specified list or combo box (as opposed to the field that is displayed). Applies only to list boxes and combo boxes. Read/write **String**.

*expression*.**ListBoundField**

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

## Remarks

Use the [ListDisplayField](#) property to return or set the displayed field.

This property is used with the [ListRowSource](#) and [ListDisplayField](#) properties to control how data is displayed in a list box or combo box. For example, consider a list box that displays product names in a section called Order Details. The recordset behind the Order Details section includes a field named ProductID that represents the product for a given Order Detail record. But instead of displaying ProductID in a textbox, the page author wants to display product names in a dropdown list box. The list box can be filled with a recordset definition named Products which contains fields called ProdID and ProductName from the Product table. The properties of this list box element would be set as shown in the following table.

Property	Value
<b>ListRowSource</b>	Products
<b>ListBoundField</b>	ProdID
<b>ListDisplayField</b>	ProductName
<b>ControlSource</b>	ProductID

# ListDisplayField Property

-

Returns or sets the field that is displayed in the list or combo box (as opposed to the field that is bound). There can be only one displayed field. This property applies only to list boxes and combo boxes. Read/write **String**.

*expression*.**ListDisplayField**

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



## Remarks

Use the [ListBoundField](#) property to return or set the bound field.

This property is used with the [ListRowSource](#) and [ListBoundField](#) properties to control how data is displayed in a list box or combo box. For example, consider a list box that displays product names in a section called Order Details. The recordset behind the Order Details section includes a field named ProductID that represents the product for a given Order Detail record. But instead of displaying ProductID in a textbox, the page author wants to display product names in a dropdown list box. The list box can be filled with a recordset definition named Products which contains fields called ProdID and ProductName from the Product table. The properties of this list box element would be set as shown in the following table.

Property	Value
<b>ListRowSource</b>	Products
<b>ListBoundField</b>	Prod ID
<b>ListDisplayField</b>	Product Name
<b>ControlSource</b>	Product ID

# ListRowSource Property

-

Specifies the data source for a list box or combo box. This property applies only to list boxes and combo boxes, and it must be set before the [ListBoundField](#) or [ListDisplayField](#) property is set. Read/write **String**.

*expression*.**ListRowSource**

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

## Remarks

This property is used with the [ListBoundField](#) and [ListDisplayField](#) properties to control how data is displayed in a list box or combo box. For example, consider a list box that displays product names in a section called Order Details. The recordset behind the Order Details section includes a field named ProductID that represents the product for a given Order Detail record. But instead of displaying ProductID in a textbox, the page author wants to display product names in a dropdown list box. The list box can be filled with a recordset definition named Products which contains fields called ProdID and ProductName from the Product table. The properties of this list box element would be set as shown in the following table.

Property	Value
<b>ListRowSource</b>	Products
<b>ListBoundField</b>	Prod ID
<b>ListDisplayField</b>	Product Name
<b>ControlSource</b>	Product ID

▼ [Show All](#)

# Location Property

-

Returns or sets the location of the calculated field evaluation. Read/write [DscLocationEnum](#).

DscLocationEnum can be one of these DscLocationEnum constants.

**dscClient**

**dscServer**

**dscSystem**

*expression*.**Location**

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

## Remarks

This property controls where calculated fields are evaluated. When this property is set to **dscServer**, the expression is included in the SQL string sent to the database and must be in the correct SQL syntax for the database. When this property is set to **dscClient**, the expression is evaluated on the client computer using Visual Basic for Applications expressions.

# Locked Property

-  
**True** if all cells in the specified range are locked, **False** if none of the cells are locked, and **Null** if some cells are locked and some are not. The default value is **True**. Use the **IsNull** function to determine whether the return value is **Null**.  
Read/write **Variant**.

*expression*.**Locked**

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

## Example

This example locks only the cells in column B on the active sheet in Spreadsheet1 and then protects the worksheet.

```
Sub LockColumnB()  
    Dim shtCurrent  
  
    Set shtCurrent = Spreadsheet1.ActiveSheet  
  
    ' Clear the locked attribute for all cells on the active sheet.  
    shtCurrent.Cells.Locked = False  
  
    ' Lock all of the cells in Column B.  
    shtCurrent.Columns(2).Locked = True  
  
    ' Enable protection on the active sheet.  
    shtCurrent.Protectio.Enabled = True  
End Sub
```





# LogBase Property

-

Sets the base of the logarithm when you are using log scales. Attempting to set this property to a value less than or equal to zero (0) causes an error. Read/write **Double**.

*expression*.**LogBase**

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

## Example

This example causes the specified axis to use a base 2 logarithmic scale.

```
Sub SetScaling()  
  
    Dim chConstants  
    Dim scValueAxisScaling  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the scaling object of the value axis.  
    Set scValueAxisScaling = ChartSpace1.Charts(0).Axes(chConstants.c  
  
    ' Set the scaling type.  
    scValueAxisScaling.Type = chConstants.chScaleTypeLogarithmic  
  
    ' Set the base value.  
    scValueAxisScaling.LogBase = 2  
  
End Sub
```



# LookupRelationships Property

-  
Returns the [LookupRelationships](#) collection for the specified page row source. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

*expression*.**LookupRelationships**

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

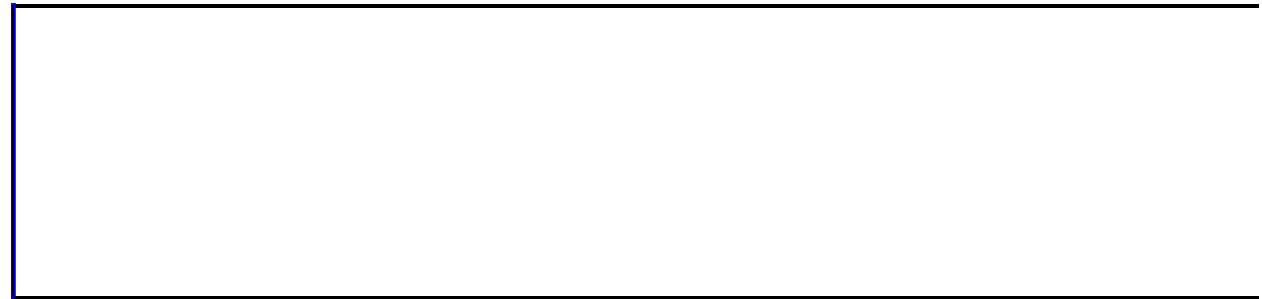
## Remarks

**LookupRelationship** objects refer to row sources that have a one-to-many relationship with the specified page row source. Fields from these row sources can appear in the same section or PivotTable list as fields from the specified row source.

# LookupSchemaRelationships Property

Returns the [LookupSchemaRelationships](#) collection for the specified schema row source. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).



*expression*.**LookupSchemaRelationships**

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

# MajorGridlines Property

-

Returns a **ChGridlines** object that represents the major gridlines for the specified axis. Note that you can use gridlines on any axis. Read-only.

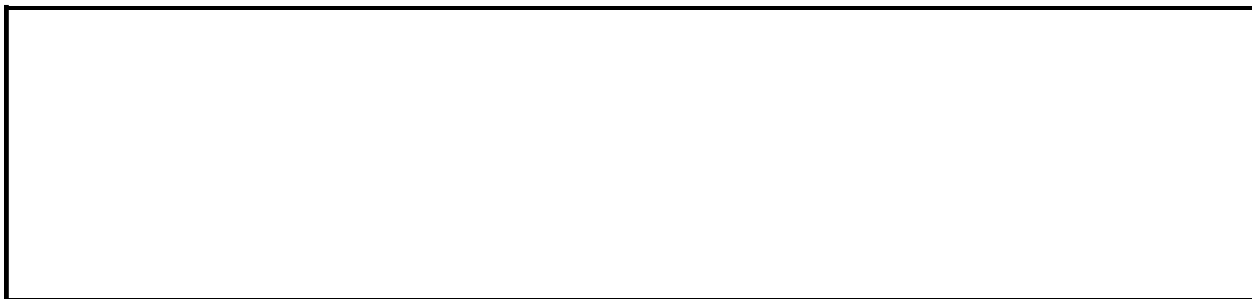
*expression*.**MajorGridlines**

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

## Example

This example sets the color and line weight for the gridlines on the value axis of the first chart in ChartSpace1.

```
Sub Format_Gridlines()  
    Dim chConstants  
    Dim glMajorGridlines  
    Dim glMinorGridlines  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the major gridlines on the value axis.  
    Set glMajorGridlines = ChartSpace1.Charts(0).Axes( _  
        chConstants.chAxisPositionValue).MajorGridlines  
  
    ' Set a variable to the minor gridlines on the value axis.  
    Set glMinorGridlines = ChartSpace1.Charts(0).Axes( _  
        chConstants.chAxisPositionValue).MinorGridlines  
  
    ' Set the color and weight of the major gridlines.  
    glMajorGridlines.Line.Color = "white"  
    glMajorGridlines.Line.Weight = 5  
  
    ' Set the color and weight of the minor gridlines.  
    glMinorGridlines.Line.Color = "yellow"  
    glMajorGridlines.Line.Weight = 2  
End Sub
```



▼ [Show All](#)



# MajorTickMarks Property

-  
Returns or sets the major tick-mark type for the specified axis. Read/write [ChartTickMarkEnum](#).

ChartTickMarkEnum can be one of these ChartTickMarkEnum constants.

**chTickMarkAutomatic**

**chTickMarkCross**

**chTickMarkInside**

**chTickMarkNone**

**chTickMarkOutside**

*expression*.**MajorTickMarks**

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

## Example

This example turns off major tick marks on the specified axis.

```
Sub DisableMajorTickMarks()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ChartSpace1.Charts(0).Axes(chConstants.chAxisPositionLeft).MajorT  
        = chConstants.chTickMarkNone  
End Sub
```



# MajorUnit Property

-

Returns or sets the major unit for the specified axis. Use this property only with a value axis. Read/write **Double**.

*expression*.**MajorUnit**

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

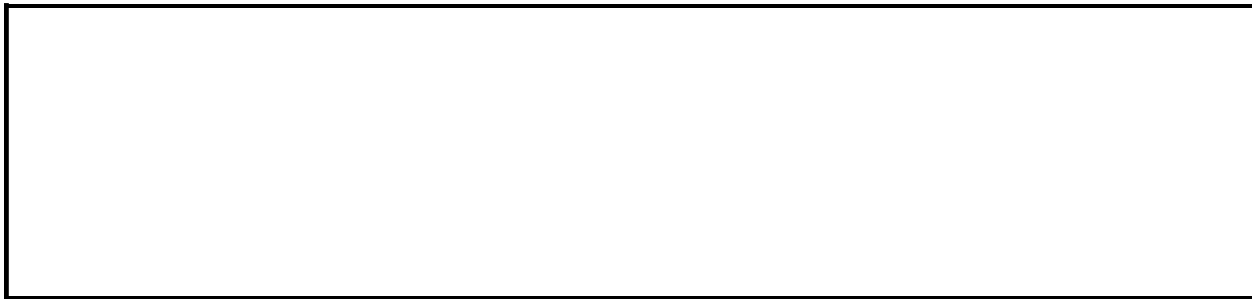
## Remarks

Setting this property causes the **HasAutoMajorUnit** property to be set to **False**.

## Example

This example sets the major and minor unit for the value axis.

```
Sub SetValueAxis()  
    Dim chConstants  
    Dim axValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo  
  
    axValueAxis.MajorUnit = 7  
    axValueAxis.MinorUnit = 2.5  
End Sub
```



# MajorVersion Property

-

Returns the major version of the Microsoft Office Web Components object library. Read-only **Long**.

---

*expression*.**MajorVersion**

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

## Example

This example displays the major version of Spreadsheet1.

Msgbox Spreadsheet1.**MajorVersion**

▼ [Show All](#)



# ManySide Property

▸ [ManySide property as it applies to the \*\*PageRelatedField\*\* object.](#)

Returns a **PageField** object that represents the field corresponding to the page row source on the many side of the specified one-to-many relationship.

*expression*.**ManySide**

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

▸ [ManySide property as it applies to the \*\*PageRelationship\*\* object.](#)

Returns a **PageRowsource** object that represents the page row source on the many side of the specified one-to-many relationship.

*expression*.**ManySide**

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

▸ [ManySide property as it applies to the \*\*SchemaRelatedField\*\* and \*\*SchemaRelationship\*\* objects.](#)

Returns the name of the schema field or schema row source on the many side of the specified one-to-many relationship. Read-only **String**.

*expression*.**ManySide**

*expression* Required. An expression that returns one of the above objects.



# Marker Property

-

Returns a **ChMarker** object that represents the markers for every point in the specified series. Read-only.

*expression*.**Marker**

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

## Example

This example sets the chart type and then sets the marker type and interior color for the specified series.

```
Sub SetMarkerStyle()  
    Dim chConstants  
    Dim serSeries1  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ChartSpace1.Charts(0).Type = chConstants.chChartTypeLineMarkers  
  
    serSeries1.Marker.Style = chConstants.chMarkerStyleSquare  
  
    serSeries1.Interior.Color = "blue"  
End Sub
```



▼ [Show All](#)

# MaxHeight Property

-

## ► [MaxHeight property as it applies to the \*\*PivotTable\*\* object.](#)

Returns or sets the maximum height in pixels that the specified PivotTable list can attain when the value of the **AutoFit** property is **True**. A vertical scroll bar is displayed if there is more data available that cannot be displayed within the maximum height. The default value is 32,000 pixels. Read/write **Long**.

*expression*.**MaxHeight**

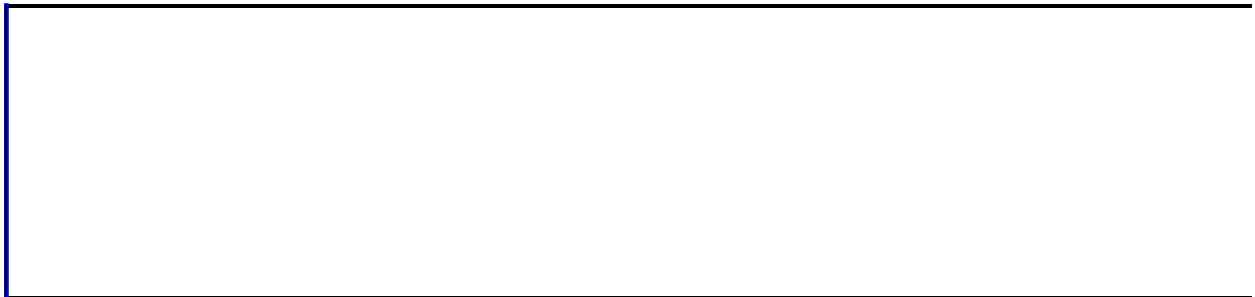
*expression* Required. An expression that returns a **PivotTable** object.

## ► [MaxHeight property as it applies to the \*\*Spreadsheet\*\* object.](#)

Returns or sets the maximum height that the specified spreadsheet can attain when the value of the **AutoFit** property is **True**. A vertical scroll bar is displayed if there is more data available that cannot be displayed within the maximum height. The value is in pixels or a percentage of the container window height (expressed as a string such as "50%"). The default value is 80%. Read/write **Variant**.

*expression*.**MaxHeight**

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



# Maximum Property

-  
Returns or sets the maximum value for the specified scale. Read/write **Double**.

*expression*.**Maximum**

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

## Example

This example sets the maximum and minimum values for the specified axis scale.

```
Sub SetScalingLimits()  
    Dim chConstants  
    Dim axisScale  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set axisScale = ChartSpace1.Charts(0).Axes(chConstants.chAxisPosi  
  
    axisScale.Maximum = 70  
    axisScale.Minimum = -10  
End Sub
```





# MaxRecords Property

-

Returns or sets the maximum number of records that the connection will return to the local computer. Read/write **Long**.

---

*expression*.**MaxRecords**

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

▼ [Show All](#)

# MaxWidth Property

## ▸ [MaxWidth property as it applies to the \*\*PivotTable\*\* object.](#)

Returns or sets the maximum width in pixels that the specified PivotTable list can attain when the value of the **AutoFit** property is **True**. A horizontal scroll bar is displayed if there is more data available that cannot be displayed within the maximum width. The default value is 32,000 pixels. Read/write **Long**.

*expression*.**MaxWidth**

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

## ▸ [MaxWidth property as it applies to the \*\*Spreadsheet\*\* object.](#)

Returns or sets the maximum width that the specified spreadsheet can attain when the value of the **AutoFit** property is **True**. A horizontal scroll bar is displayed if there is more data available that cannot be displayed within the maximum width. The value is in pixels or a percentage of the container window width (expressed as a string such as "50%"). The default value is 80%. Read/write **Variant**.

*expression*.**MaxWidth**

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

---

▼ [Show All](#)

# Member Property

▸ [Member property as it applies to the \*\*PivotResultColumnAxis\*\*, \*\*PivotResultGroupAxis\*\*, \*\*PivotResultPageAxis\*\*, and \*\*PivotResultRowAxis\*\* objects.](#)

Returns a **PivotAxisMember** object that represents the parent member of the specified result axis.

*expression*.Member

*expression* Required. An expression that returns one of the above objects.

▸ [Member property as it applies to the \*\*PivotFieldSet\*\* object.](#)

Returns a **PivotMember** object that represents the top member object for the specified field set.

*expression*.Member

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



# MemberCaptions Property

-

Returns or sets an array of **Variant** values that contains the captions of the members in the specified field. Use this property to customize the captions of the members in a field. Read/write.

*expression*.**MemberCaptions**

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

## Remarks

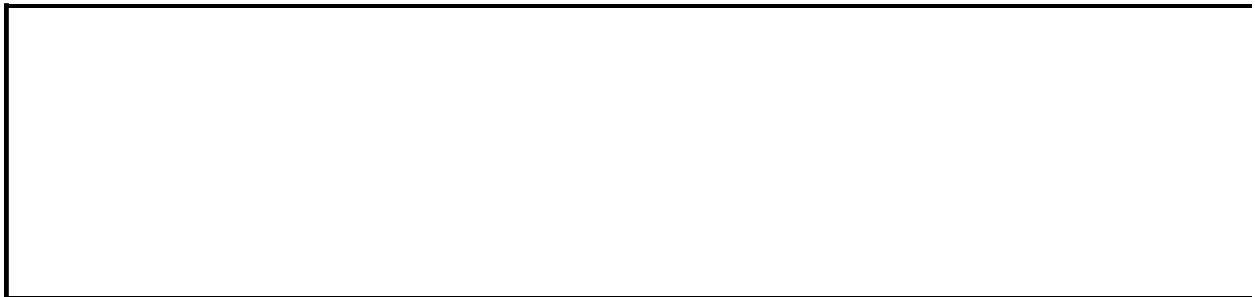
The array that you pass to this property contains an array for each caption that you want to modify. The first element in the array can contain either a member name, unique name, or a reference to a **PivotMember** object. The second element in the array is the new caption to be used for the member.

Members not specified in the array will use the default captions provided by the data source.

## Example

This example replaces the captions in the State Province field of the Customers field set with captions that are more readable.

```
Sub NewMemberCaptions()  
  
    Dim fldStateCaptions  
    Dim avarNewCaptions(2)  
  
    Set fldStateCaptions = PivotTable1.ActiveView.FieldSets("Customers"  
        .Fields("State Province"))  
  
    ' The following three lines of code specify the new captions to  
    ' be displayed for the states in the State Province field.  
    avarNewCaptions(0) = Array("[State Province].[CA]", "California")  
    avarNewCaptions(1) = Array("[State Province].[WA]", "Washington")  
    avarNewCaptions(2) = Array("[State Province].[OR]", "Oregon")  
  
    ' Apply the new captions to the State Province field.  
    fldStateCaptions.MemberCaptions = avarNewCaptions  
  
End Sub
```





▼ [Show All](#)

# MemberProperties Property

▸ [MemberProperties property as it applies to the \*\*PivotField\*\* object.](#)

Returns a **PivotMemberProperties** object that represents the OLAP member properties for the specified PivotTable field.

*expression*.**MemberProperties**

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

▸ [MemberProperties property as it applies to the \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotPageMember\*\*, and \*\*PivotRowMember\*\* objects.](#)

Returns a **PivotResultMemberProperties** object that represents the OLAP member properties for the specified PivotTable field.

*expression*.**MemberProperties**

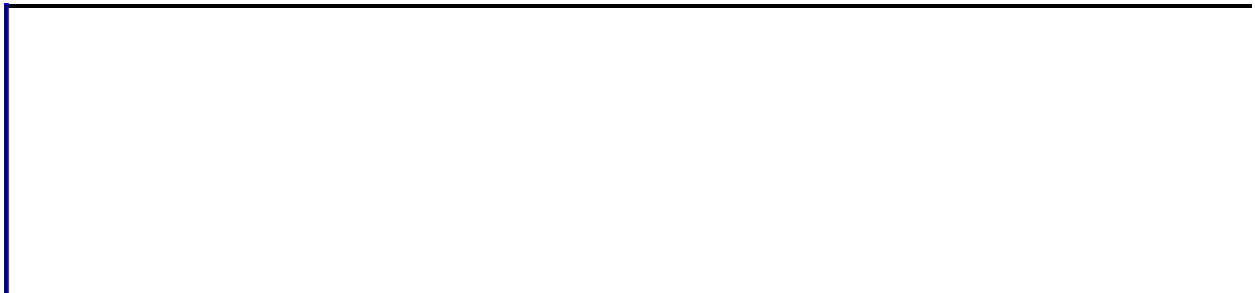
*expression* Required. An expression that returns one of the above objects.

## Example

► [As it applies to the \*\*PivotField\*\* object.](#)

This example displays the member properties for the Store Name field set.

```
Sub Display_MemberProperties()  
  
    Dim ptView  
    Dim ptConstants  
    Dim fldStoreName  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view of the PivotTable.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Store Name field.  
    Set fldStoreName = ptView.FieldSets("Store").Fields("Store Name")  
  
    ' The following three lines of code specify that the member property  
    ' displayed in the PivotTable list.  
    fldStoreName.MemberProperties("Store Manager").DisplayIn = ptConstants  
    fldStoreName.MemberProperties("Store Type").DisplayIn = ptConstants  
    fldStoreName.MemberProperties("Store Sqft").DisplayIn = ptConstants  
  
    ' The following three lines of code set the caption for the member  
    fldStoreName.MemberProperties("Store Manager").Caption = "Manager N  
    fldStoreName.MemberProperties("Store Type").Caption = "Store Type"  
    fldStoreName.MemberProperties("Store Sqft").Caption = "Size in SQFT  
  
End Sub
```



# MemberPropertiesOrder Property

-

Returns or sets an array of **Variant** values that represents the order that the member properties are to be displayed in the specified field. Use this property to rearrange the order that member properties are displayed in. Read/write.

*expression*.**MemberPropertiesOrder**

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

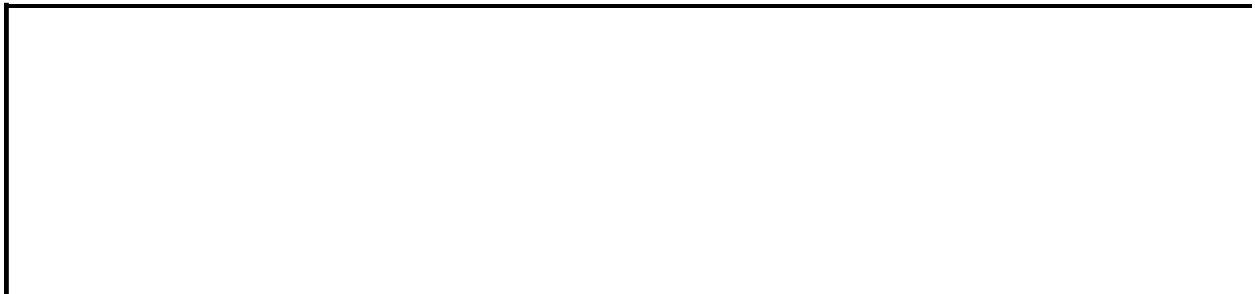
## Remarks

The array that you pass to this property can contain a list of **PivotMemberProperty** objects or a **String** list of member captions.

## Example

This example enables the member captions for the Store Name field in PivotTable1. Then, the member captions are rearranged, and their captions are customized.

```
Sub Format_MemberProperties()  
    Dim ptView  
    Dim ptConstants  
    Dim fldStoreName  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the active view of the PivotTable.  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Store Name field.  
    Set fldStoreName = ptView.FieldSets("Store").Fields("Store Name")  
  
    ' The following three lines of code specify that the member prop  
    ' displayed in the PivotTable list.  
    fldStoreName.MemberProperties("Store Manager").DisplayIn = ptCon  
    fldStoreName.MemberProperties("Store Type").DisplayIn = ptConsta  
    fldStoreName.MemberProperties("Store Sqft").DisplayIn = ptConsta  
  
    fldStoreName.MemberPropertiesOrder = Array("Store Type", "Store  
  
    ' The following three lines of code set the caption for the memb  
    fldStoreName.MemberProperties("Store Manager").Caption = "Manage  
    fldStoreName.MemberProperties("Store Type").Caption = "Store Typ  
    fldStoreName.MemberProperties("Store Sqft").Caption = "Size in S  
End Sub
```



# MemberProperty Property

-

Returns a **PivotMemberProperty** object that represents the member property of the specified result member.

*expression*.**MemberProperty**

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



# MergeArea Property

-

Returns a [Range](#) object that represents the merged range containing any part of the specified range. The specified range can contain more than one cell; if the range does not contain any merged cells, it is returned unchanged. Read-only.

*expression*.**MergeArea**

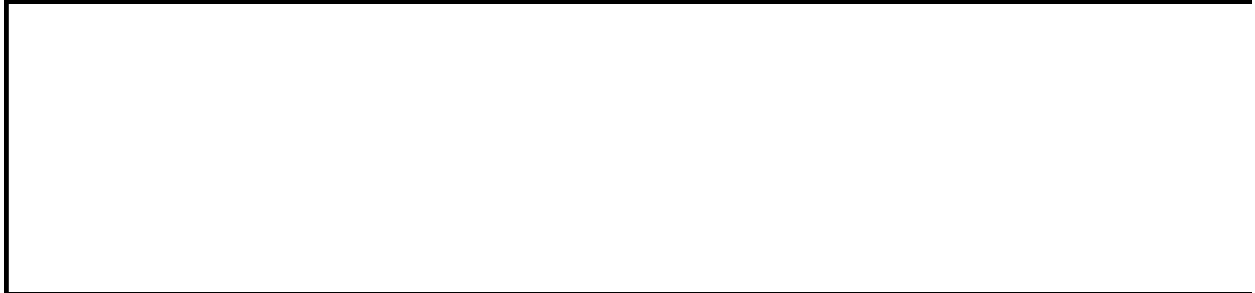
*expression* Required. An expression that returns a **Range** object.



## Example

This example creates a merged cell from the range B2:C5 in Sheet1 and puts a thick red border around the merged cell.

```
Sub Merge_Cells()  
    Dim ssConstants  
    Dim shtCurrent  
  
    Set ssConstants = Spreadsheet1.Constants  
    Set shtCurrent = Spreadsheet1.Worksheets("Sheet1")  
  
    ' Merge cells B2:C5.  
    shtCurrent.Range("B2:C5").Merge  
  
    ' Set the border color of the merged cell.  
    shtCurrent.Range("B2").MergeArea.Borders.Color = "Red"  
  
    ' Set the border weight of the merged cell.  
    shtCurrent.Range("B2").MergeArea.Borders.Weight = ssConstants.ow  
End Sub
```



# MergeCells Property

**True** if the specified range contains only merged cells. Read/write **Variant**.

*expression*.**MergeCells**

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

## 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.

Use the **UnMerge** method to unmerge the specified range.

--

# Minimum Property

-  
Returns or sets the minimum value for the specified scale. Read/write **Double**.

*expression*.**Minimum**

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

## Example

This example sets the maximum and minimum values for the specified axis scale.

```
Sub SetScalingLimits()  
    Dim chConstants  
    Dim axisScale  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set axisScale = ChartSpace1.Charts(0).Axes(chConstants.chAxisPosi  
  
    axisScale.Maximum = 70  
    axisScale.Minimum = -10  
End Sub
```



# MinorGridlines Property

-

Returns a **ChGridlines** object that represents the minor gridlines for the specified axis. Note that you can use gridlines on any axis. Read-only.

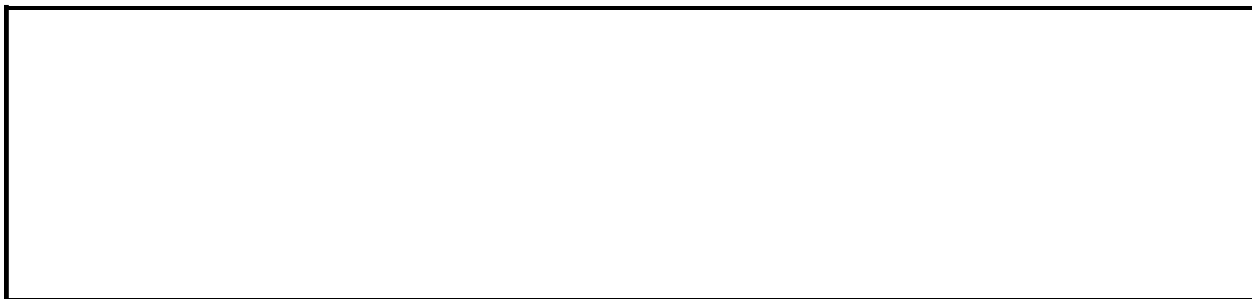
*expression*.**MinorGridlines**

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

## Example

This example sets the color and line weight for the gridlines on the value axis of the first chart in ChartSpace1.

```
Sub Format_Gridlines()  
    Dim chConstants  
    Dim glMajorGridlines  
    Dim glMinorGridlines  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the major gridlines on the value axis.  
    Set glMajorGridlines = ChartSpace1.Charts(0).Axes( _  
        chConstants.chAxisPositionValue).MajorGridlines  
  
    ' Set a variable to the minor gridlines on the value axis.  
    Set glMinorGridlines = ChartSpace1.Charts(0).Axes( _  
        chConstants.chAxisPositionValue).MinorGridlines  
  
    ' Set the color and weight of the major gridlines.  
    glMajorGridlines.Line.Color = "white"  
    glMajorGridlines.Line.Weight = 5  
  
    ' Set the color and weight of the minor gridlines.  
    glMinorGridlines.Line.Color = "yellow"  
    glMinorGridlines.Line.Weight = 2  
End Sub
```



▼ [Show All](#)



# MinorTickMarks Property

-  
Returns or sets the minor tick-mark type for the specified axis. Read/write [ChartTickMarkEnum](#).

ChartTickMarkEnum can be one of these ChartTickMarkEnum constants.

**chTickMarkAutomatic**

**chTickMarkCross**

**chTickMarkInside**

**chTickMarkNone**

**chTickMarkOutside**

*expression*.**MinorTickMarks**

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

## Example

This example turns on minor gridlines and minor tick marks for the specified axis and sets the gridline color and tick-mark style.

```
Sub FormatValueAxis()  
    Dim chConstants  
    Dim axValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo  
    axValueAxis.HasMinorGridlines = True  
  
    axValueAxis.MinorGridlines.Line.Color = "green"  
  
    axValueAxis.MinorTickMarks = chConstants.chTickMarkOutside  
End Sub
```



# MinorUnit Property

-

Returns or sets the minor unit for the specified axis. Use this property only with a value axis. Read/write **Double**.

*expression*.**MinorUnit**

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

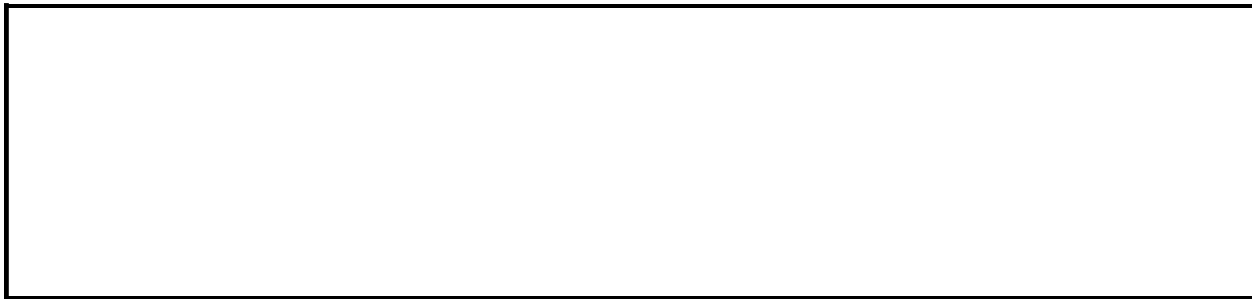
## Remarks

Setting this property causes the **HasAutoMinorUnit** property to be set to **False**.

## Example

This example sets the major and minor unit for the value axis.

```
Sub SetValueAxis()  
    Dim chConstants  
    Dim axValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    Set axValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo  
  
    axValueAxis.MajorUnit = 7  
    axValueAxis.MinorUnit = 2.5  
End Sub
```



# MinorVersion Property

-

Returns the minor version of the Microsoft Office Web Components object library. Read-only **String**.

---

*expression*.**MinorVersion**

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

▼ [Show All](#)

# Miter Property

-  
Returns or sets a [ChartLineMiterEnum](#) constant indicating the way that a line's exterior edges are joined. Read/write.

ChartLineMiterEnum can be one of these ChartLineMiterEnum constants.

**chLineMiterBevel** *default*

**chLineMiterMiter**

**chLineMiterRound**

*expression*.**Miter**

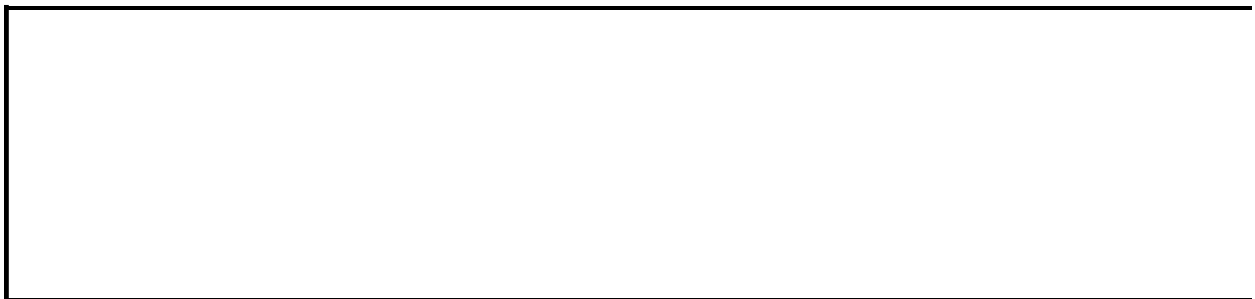
*expression* Required. An expression that returns a **ChLine** object.



## Example

This example changes the first chart in Chartspace1 to a line chart and then formats the line for the first data series in the chart.

```
Sub Set_Series_LineStyle()  
  
    Dim chConstants  
    Dim serSeries1  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Change the chart to a line chart.  
    ChartSpace1.Charts(0).Type = chChartTypeLine  
  
    ' Set a variable to refer to the first data series in the chart.  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Set the miter of the line of the first series.  
    serSeries1.Line.Miter = chConstants.chLineMiterBevel  
  
    ' Set the line weight of the first series.  
    serSeries1.Line.Weight = chConstants.owcLineWeightThick  
  
    ' Set the line style of the first series.  
    serSeries1.Line.DashStyle = chConstants.chLineRoundDot  
  
End Sub
```



# MoveAfterReturn Property

-

Determines whether the active cell will be moved when the user presses the ENTER key. The default value is **True**. Use the **MoveAfterReturnDirection** property to determine which direction the active cell moves when the user presses ENTER. Read/write **Boolean**.

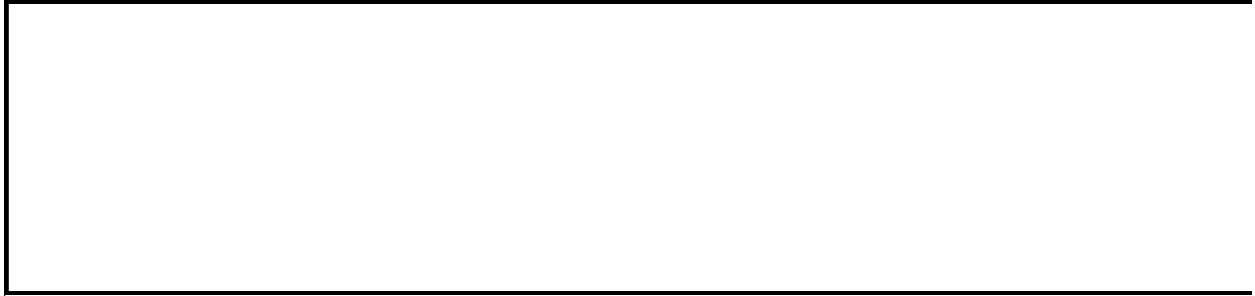
*expression*.**MoveAfterReturn**

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

## Example

This example causes the active cell to remain selected after the user presses the ENTER key.

```
Spreadsheet1.MoveAfterReturn = False
```



# MoveAfterReturnDirection Property

-

Returns or sets the direction in which the focus is moved when the user presses the ENTER key and the **MoveAfterReturn** property is set to **True**. Read/write [XlDirection](#).

XlDirection can be one of these XlDirection constants.

**xlDown**

**xlToLeft**

**xlToRight**

**xlUp**

*expression*.**MoveAfterReturnDirection**

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

## Example

This example causes the cell to the right of the active cell to be selected after the user presses the ENTER key.

```
Sub MoveAfterEnter()  
    Dim ssConstants  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    ' Enable the MoveAfterReturn property.  
    Spreadsheet1.MoveAfterReturn = True  
  
    ' Move the cursor one cell to the right when  
    ' the user presses ENTER.  
    Spreadsheet1.MoveAfterReturnDirection = xlToRight  
End Sub
```



# Name Property

-

Returns or sets the name of the specified object.

- Read/write **Variant** for the **Font** object.
- Read/write **Name** object for the **Range** object.
- Read/write **String** for the **ChAxis**, **ChChart**, **ChFont**, **ChSeries**, **GroupingDef**, **Name**, **PageField**, **PivotFont**, **PivotHyperlink**, **PivotTotal**, **RecordsetDef**, **SchemaField**, **SchemaRowsource**, and **Worksheet** objects.
- Read-only **String** for all other objects in the Applies To list.

*expression*.**Name**

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

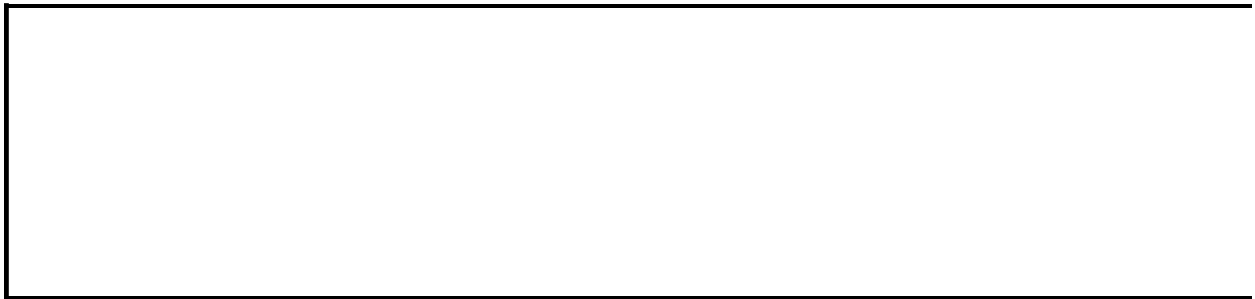
## Example

This example sets the name of the specified chart.

```
ChartSpace1.Charts(0).Name = "Line Chart"
```

After the chart name has been set, you can refer to the chart by name, as shown in the following example.

```
ChartSpace1.Charts("Line Chart").HasLegend = True
```



▼ [Show All](#)



# Names Property

## ▸ [Names property as it applies to the \*\*Spreadsheet\*\* and \*\*Workbook\*\* objects.](#)

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

*expression*.**Names**

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

## ▸ [Names property as it applies to the \*\*Worksheet\*\* object.](#)

Returns a **Names** collection that represents all the worksheet-level names that are defined in the specified worksheet. Read-only.

*expression*.**Names**

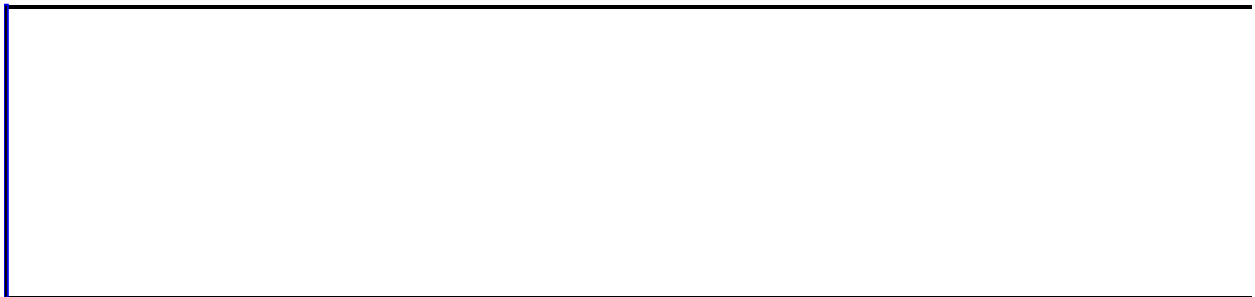
*expression* Required. An expression that returns a **Worksheet** object.

## Example

► [As it applies to the \*\*Spreadsheet\*\* and \*\*Workbook\*\* objects.](#)

The following example creates a list of all the names in the active workbook, along with the addresses to which they refer.

```
Sub List_All_Names()  
    Dim nmCurrentName  
    Dim rngCurrent  
  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Range("A1")  
  
    ' Loop through all of the names in the active workbook.  
    For Each nmCurrentName In Spreadsheet1.ActiveWorkbook.Names  
  
        ' Write the current name to the worksheet.  
        rngCurrent.Value = nmCurrentName.Name  
  
        ' Write the definition of the current name to the worksheet.  
        rngCurrent.Offset(0, 1).Value = "'" & nmCurrentName.RefersTo  
  
        Set rngCurrent = rngCurrent.Offset(1, 0)  
    Next  
End Sub
```



▼ [Show All](#)

# Next Property

## ▶ [Next property as it applies to the \*\*Range\*\* object.](#)

Returns a **Range** object that represents the next cell. This property emulates pressing TAB; unlike the keystroke, the property returns the previous 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.

*expression*.**Next**

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

## ▶ [Next property as it applies to the \*\*Worksheet\*\* object.](#)

Returns a **Worksheet** object that represents the next sheet.

*expression*.**Next**

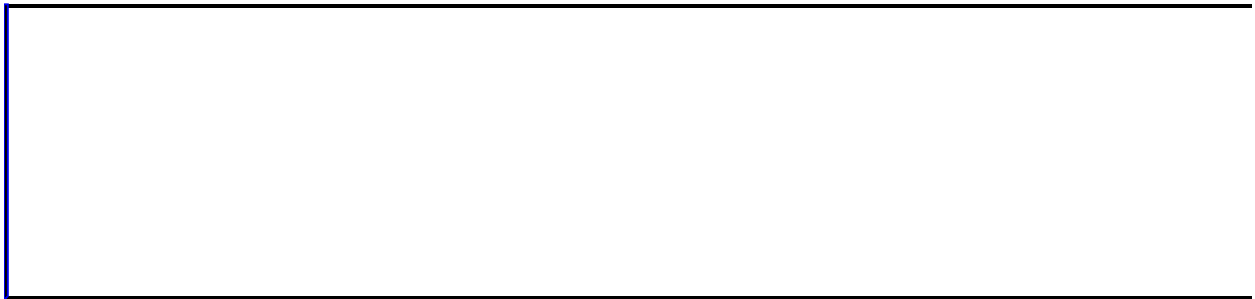
*expression* Required. An expression that returns a **Worksheet** object.

## Example

► [As it applies to the \*\*Range\*\* object.](#)

This example selects the next unlocked cell on the active worksheet. If the active worksheet is unprotected, this is the cell immediately to the right of the active cell.

```
Spreadsheet1.ActiveCell.Next.Select
```



# NextSection Property

-

Returns a [Section](#) object that represents the next physical section on the specified page, regardless of siblings or parents. This property fails on the final section on the page. Read-only.

---

*expression*.**NextSection**

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

# NextSibling Property

-

Returns a [Section](#) object that represents the next sibling in the current data access page. This may cause the next page of records to populate itself. This property will not cross parents, and it will fail on the last sibling of the current data access page.

*expression*.**NextSibling**

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

---

---

# NumberFormat Property

-

Returns or sets the number format for the specified object. Read/write **Variant** for the [Range](#) object; read/write **String** for all other objects in the Applies To list.

*expression*.**NumberFormat**

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



## Example

The following example sets the number format for the ranges A1:E10 and F1:F10 on the active worksheet.

```
Sub FormatCells()  
    Spreadsheet1.ActiveSheet.Range("A1:E10").NumberFormat = "0.###"  
    Spreadsheet1.ActiveSheet.Range("F1:F10").NumberFormat = "Currency"  
End Sub
```



# NumericScale Property

-

Returns the maximum number of digits you can have to the right of the decimal point for numeric types. Read-only **Long**.

*expression*.**NumericScale**

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

--

▼ [Show All](#)

# ObjectType Property

-

Returns a [ChartSelectionsEnum](#) that represents the type of object that is currently selected.

ChartSelectionsEnum can be one of these ChartSelectionsEnum constants.

**chSelectionAxis**

**chSelectionCategoryLabel**

**chSelectionChart**

**chSelectionChartSpace**

**chSelectionDataLabel**

**chSelectionDataLabels**

**chSelectionDropZone**

**chSelectionErrorbars**

**chSelectionField**

**chSelectionGridlines**

**chSelectionLegend**

**chSelectionLegendEntry**

**chSelectionNone**

**chSelectionPlotArea**

**chSelectionPoint**

**chSelectionSeries**

**chSelectionSurface**

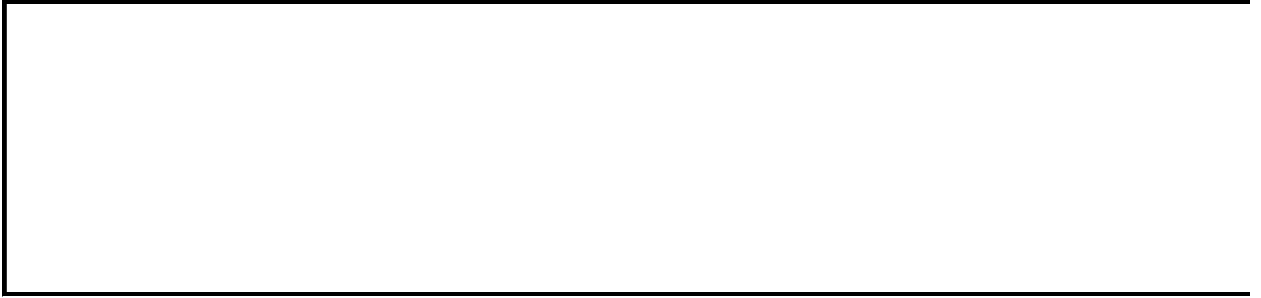
**chSelectionTitle**

**chSelectionTrendline**

**chSelectionUserDefined**

*expression*.**ObjectType**

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



# Offline Property

-  
Returns a **Boolean** indicating whether Microsoft Internet Explorer is in offline mode. Read-only.

*expression*.**Offline**

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

## Example

This example changes the text of a label control to indicate the offline status of a data access page before the data access page is bound to its data source.

```
Sub MSODSC_BeforeInitialBind(DSCEventInfo)
```

```
    If MSODSC.Offline = True then  
        Label.innerText = "Offline"  
    Else  
        Label.innerText = "Online"  
    End If
```

```
End Sub
```



# OfflinePublication Property

-

Returns or sets a **String** that represents the publication to use when the data access page is taken offline. Read/write.

*expression*.**OfflinePublication**

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



## Remarks

This property is valid only when the **OfflineType** property is set to **dscOfflineMerge**.

# OfflineSource Property

-

Returns or sets a **String** that represents the data source used when the data access page is offline. Read/write.

*expression*.**OfflineSource**

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

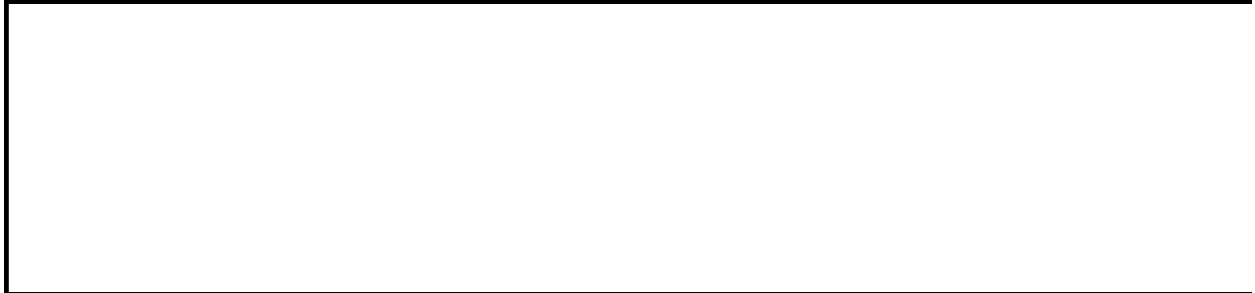
## Remarks

The value used to set this property varies based upon the current setting of the **OfflineType** and **XMLLocation** properties. If the **OfflineType** property is set to **dscOfflineMerge**, then this property should be set to a connection string for the local MSDE database. If the **OfflineType** property is set to **dscOfflineXMLDataFile** and the **XMLLocation** property is set to **dscXMLDataFile**, then this property should be set to a string containing the path to the XML data file.

## Example

This example sets the offline settings from the data source control named MSODSC.

```
Sub SetOfflineSettings()  
    Dim dscConstants  
  
    Set dscConstants = MSODSC.Constants  
  
    ' Set the offline type.  
    MSODSC.OfflineType = dscConstants.dscOfflineXMLDataFile  
  
    ' Set the location of the XML data to a data file.  
    MSODSC.XMLLocation = dscConstants.dscXMLDataFile  
  
    ' Set the XML data file to use when the page is offline.  
    MSODSC.OfflineSource = "Q1 Sales Analysis.xml"  
  
End Sub
```



▼ [Show All](#)

# OfflineType Property

-

Returns or sets a [DscOfflineTypeEnum](#) constant that represents the type of connection used to persist the data when a data access page is taken offline. Read/write.

DscOfflineTypeEnum can be one of these DscOfflineTypeEnum constants.

**dscOfflineMerge** The data is stored in a local MSDE database.

**dscOfflineNone** The data is not available when the data access page is taken offline.

**dscOfflineWorkflow** Not supported for this property.

**dscOfflineXMLDataFile** The data is stored in an XML data file.

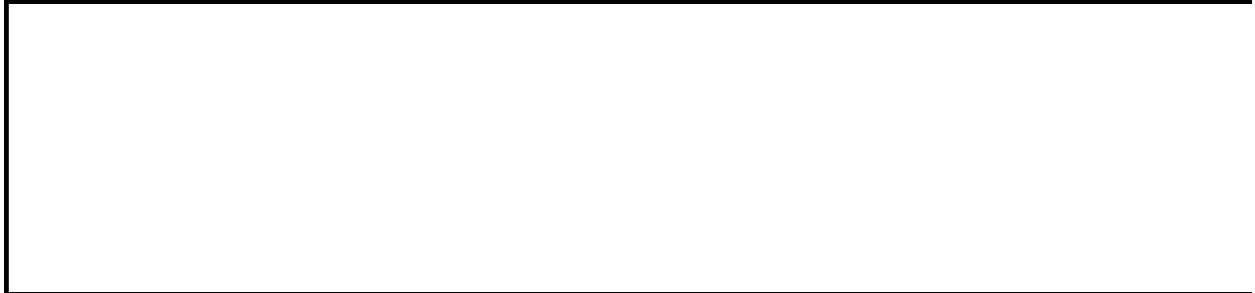
*expression*.**OfflineType**

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

## Example

This example sets the offline settings from the data source control named MSODSC.

```
Sub SetOfflineSettings()  
    Dim dscConstants  
  
    Set dscConstants = MSODSC.Constants  
  
    ' Set the offline type.  
    MSODSC.OfflineType = dscConstants.dscOfflineXMLDataFile  
  
    ' Set the location of the XML data to a data file.  
    MSODSC.XMLLocation = dscConstants.dscXMLDataFile  
  
    ' Set the XML data file to use when the page is offline.  
    MSODSC.OfflineSource = "Q1 Sales Analysis.xml"  
  
End Sub
```



# Offset Property

-  
Returns a **Range** object that represents a range that is offset from the specified range.

*expression.Offset(RowOffset, ColumnOffset)*

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

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

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



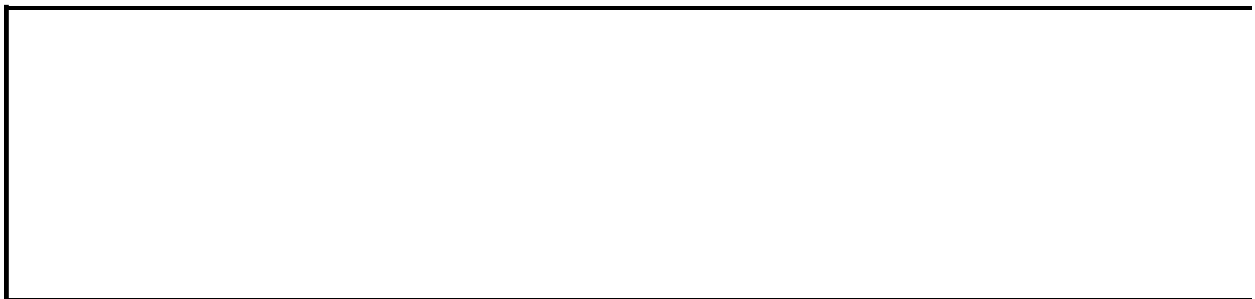
## Example

This example sets the font for the contents of the cell that is one column to the right of the active cell.

```
Spreadsheet1.ActiveCell.Offset(0, 1).Font.Bold = True
```

This example loops through the contiguous values in column A in the active sheet of Spreadsheet1 and deletes any rows that contain odd-numbered values.

```
Sub Delete_Odd_Values()  
    Spreadsheet1.ActiveSheet.Range("A1").Select  
    ' Loop until an empty cell is selected.  
    Do Until IsEmpty(Spreadsheet1.ActiveCell)  
        ' If the active cell contains an odd number.  
        If Spreadsheet1.ActiveCell.Value Mod 2 = 1 Then  
            ' Delete the row.  
            Spreadsheet1.ActiveCell.EntireRow.Delete  
        Else  
            ' Select the next cell.  
            Spreadsheet1.ActiveCell.Offset(1, 0).Select  
        End If  
    Loop  
End Sub
```



▼ [Show All](#)

# OneSide Property

## ▶ [OneSide property as it applies to the \*\*PageRelatedField\*\* object.](#)

Returns a **PageField** object that represents the field corresponding to the page row source on the “one” side of the specified one-to-many relationship.

*expression*.**OneSide**

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

## ▶ [OneSide property as it applies to the \*\*PageRelationship\*\* object.](#)

Returns a **PageRowsource** object that represents the page row source on the “one” side of the specified one-to-many relationship.

*expression*.**OneSide**

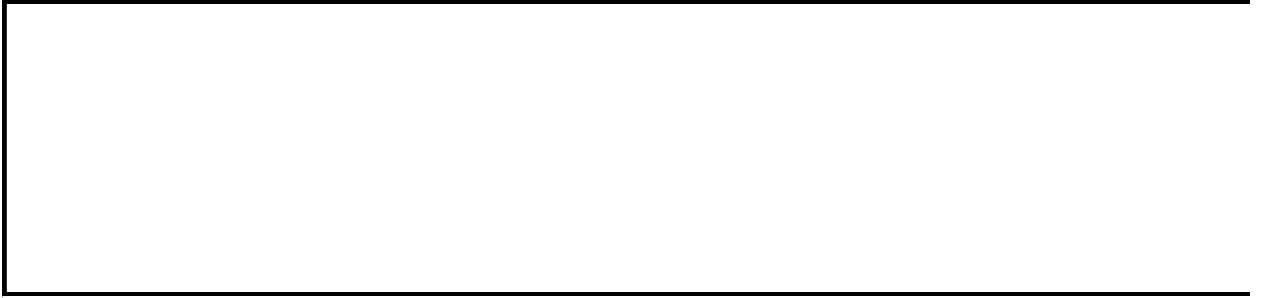
*expression* Required. An expression that returns a **PageRelationship** object.

## ▶ [OneSide property as it applies to the \*\*SchemaRelatedField\*\* and \*\*SchemaRelationship\*\* objects.](#)

Returns the name of the schema field or the schema row source on the “one” side of the specified one-to-many relationship. Read-only **String**.

*expression*.**OneSide**

*expression* Required. An expression that returns one of the above objects.



# Order Property

-

Returns or sets the polynomial order for the specified trendline (an integer value greater than 1). If you set this property for a linear trendline, it becomes a polynomial trendline. Read/write **Long**.

*expression*.**Order**

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

## Example

This example sets the order for the specified trendline.

```
ChartSpace1.Charts(0).SeriesCollection(0).Trendlines(0).Order = 2
```



# OrderedMembers Property

-

Returns or sets a **Variant** that determines how the members of a field are sorted when the **SortDirection** property is set to **plSortDirectionCustom**, **plSortDirectionCustomAscending**, or **plSortDirectionCustomDescending**. Set this property to an array of members that is ordered in the way that you want them to appear in the PivotTable list. When setting this property, you can pass an array of member names, member unique names, or member object references. When retrieving this property, the array will always contain PivotMember object references. Read/write.

*expression*.**OrderedMembers**

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

## Remarks

Any members in the field that are not listed in this array will appear below the last member listed in the array.

Setting this property replaces the current list. To add items to an existing list, you must retrieve the list and add add members to it, or use a variable to hold the current list, append to it, and reset this property.

It is allowable to pass member names or unique names that currently do not exist in the data results. If you pass a name or unique name that cannot be resolved to a resulting member, it will be converted to a PivotMember object with its **IsValid** property set to **False**.

To clear this list, set this property to **Empty** or an array of zero elements.



## Example

This example creates a custom sort order in the Title field.

```
Sub CustomSort()  
  
    Dim ptConstants  
    Dim ptView  
    Dim pfTitle  
  
    Set ptConstants = PivotTable1.Constants  
  
    Set ptView = PivotTable1.ActiveView  
  
    ' Set a variable to the Title field.  
    Set pfTitle = ptView.FieldSets("Title").Fields("Title")  
  
    ' Specify the ordering for some of the items in the Title field.  
    pfTitle.OrderedMembers = Array("Sales Representative", _  
                                    "Sales Manager", "Vice President",  
  
    ' Set the sort direction.  
    pfTitle.SortDirection = ptConstants.plSortDirectionCustom  
  
End Sub
```



▼ [Show All](#)

# Orientation Property

-

**PivotFieldSet** object: Returns a [PivotFieldSetOrientationEnum](#) constant that represents the field set orientation. A field set can be located on both the summary axis and the column, row, or filter axis at the same time. Read-only.

PivotFieldSetOrientationEnum can be one of these  
PivotFieldSetOrientationEnum constants.

**plOrientationColumnAxis**

**plOrientationDataAxis**

**plOrientationFilterAxis**

**plOrientationNone**

**plOrientationPageAxis**

**plOrientationRowAxis**

**ChAxis** object: Returns or sets a **Long** that represents the orientation of the labels on the specified axis. Can be a [ChartLabelOrientationEnum](#) constant. Read/write.

ChartLabelOrientationEnum can be one of these ChartLabelOrientationEnum constants.

**chLabelOrientationAutomatic**

**chLabelOrientationDownward**

**chLabelOrientationHorizontal**

**chLabelOrientationUpward**

**ChScaling** object: Returns or sets a [ChartScaleOrientationEnum](#) constant that represents the scaling orientation. Changing the value of this property flips the chart. Read/write.

ChartScaleOrientationEnum can be one of these ChartScaleOrientationEnum constants.

**chScaleOrientationMaxMin**  
**chScaleOrientationMinMax**

*expression*.**Orientation**

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

## Example

This example sets the scaling orientation for the specified axis.

```
Sub SetAxisOrientation()  
    Dim chConstants  
    Set chConstants = ChartSpace1.Constants  
    ChartSpace1.Charts(0).Axes(1).Scaling.Orientation = chConstants.c  
End Sub
```



# Overlap Property

-

Returns or sets the amount of overlap between markers within a single category. Positive values cause the markers to overlap, and negative values cause the markers to separate. The default value is zero (0), and the valid range is from –100 through 100. Read/write **Long**.

*expression*.**Overlap**

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

## Example

This example sets the overlap value for the specified chart.

```
ChartSpace1.Charts(0).Overlap = 25
```



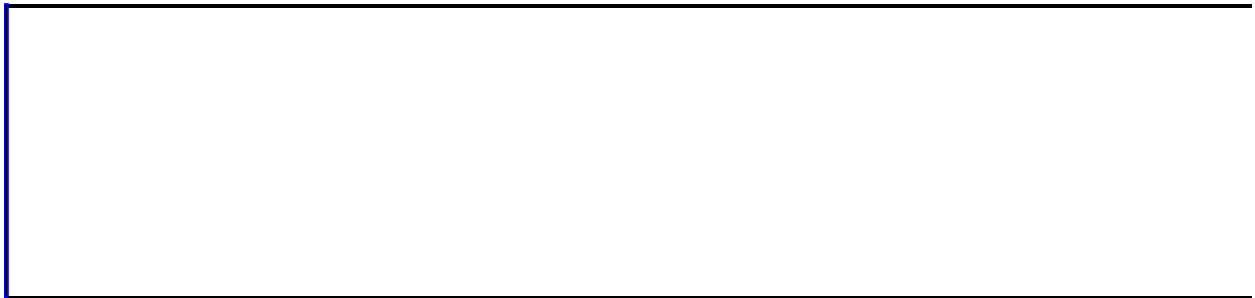
# PageAxis Property

**PivotView** object: Returns a **PivotGroupAxis** object that represents the page axis.

**PivotData** object: Returns a **PivotResultGroupAxis** object that represents the page axis.

*expression*.**PageAxis**

*expression* Required. An expression that returns a **PivotView** or **PivotData** object.





# PageFields Property

-  
Returns the [PageFields](#) collection for the specified recordset definition.

*expression*.**PageFields**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

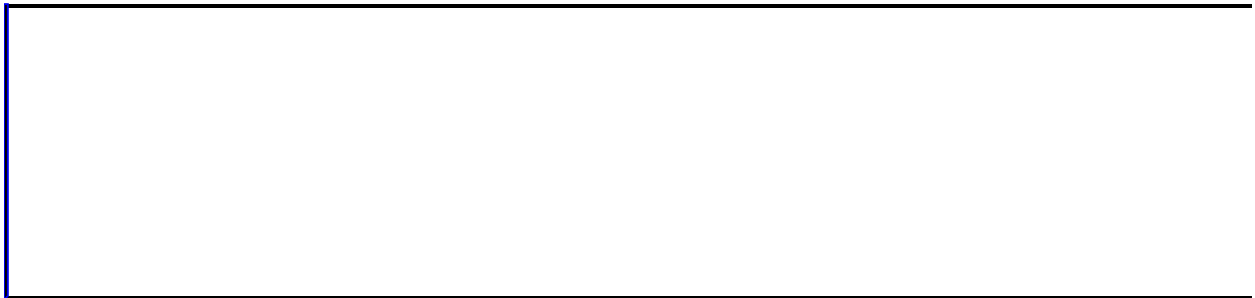
# PageMember Property

**PivotCell** object: Returns a **PivotPageMember** object that represents the page member for the specified cell.

**PivotResultPageAxis** object: Returns a **PivotPageMember** object that represents the page member for the specified result axis.

*expression*.**PageMember**

*expression* Required. An expression that returns a **PivotCell** or **PivotResultPageAxis** object.



# PageRelatedFields Property

Returns the [PageRelatedFields](#) collection for the specified page relationship.  
Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).



*expression*.**PageRelatedFields**

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

# PageRowsource Property

-

Returns the [PageRowsource](#) object for the specified page field. Read-only.



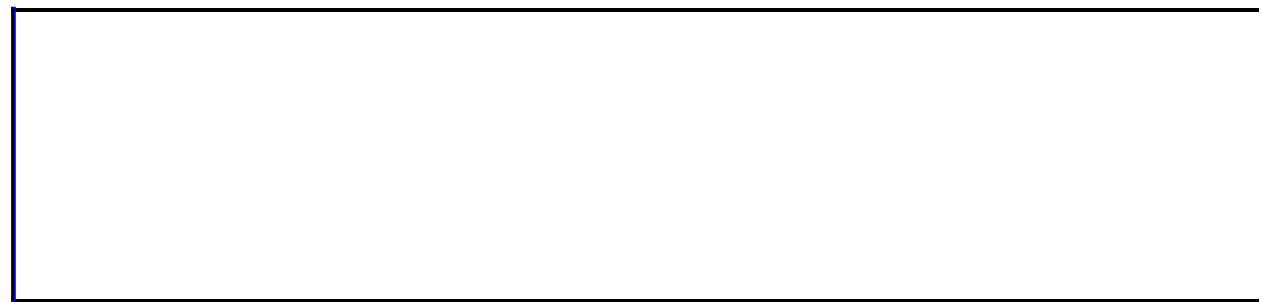
*expression*.**PageRowsource**

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

# PageRowsources Property

Returns the [PageRowsources](#) collection for the specified recordset definition.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).



*expression*.**PageRowsources**

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

# Panes Property

-  
Returns the [Panes](#) collection for the specified worksheet.

*expression*.**Panes**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).



## Example

This example sets a variable for the visible range in the specified pane and displays the range's location.

```
Sub SetVisibleRange()  
    Dim rngVisible  
    Set rngVisible = Spreadsheet1.ActiveSheet.Panes(1).VisibleRange  
    MsgBox "rngVisible.Address " & vr.Address  
End Sub
```



# ParameterValues Property

-  
Returns the [ParameterValues](#) collection for the specified recordset definition.

*expression*.**ParameterValues**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# Parent Property

-  
Returns the parent object for the specified object.

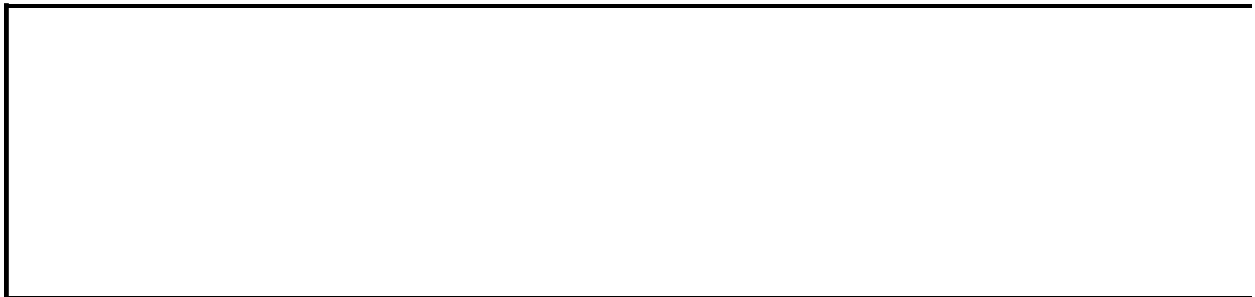
*expression*.**Parent**

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

## Example

The event procedure in this example runs whenever the user clicks in the chart workspace.

```
Sub ChartSpace1_Click()  
  
    Dim chConstants  
    Dim iSeriesNum  
    Dim iPointNum  
  
    Set chConstants = ChartSpace1.Constants  
  
    If ChartSpace1.SelectionType = chConstants.chSelectionPoint Then  
        ' Point parent is series.  
        iSeriesNum = ChartSpace1.Selection.Parent.Index  
  
        iPointNum = ChartSpace1.Selection.Index  
  
        MsgBox "Series: " & iSeriesNum & " Point: " & iPointNum  
    End If  
  
End Sub
```



# ParentAxisMember Property

-

Returns a **PivotAxisMember** object that represents the parent member for the specified axis member.

*expression*.**ParentAxisMember**

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

## Remarks

This property returns **Nothing** if this property is used for the top level axis member.

# ParentColumnMember Property

-

Returns a **PivotColumnMember** object that represents the parent member for the specified column member.

*expression*.**ParentColumnMember**

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



## Remarks

This property returns **Nothing** if this property is used for the top level column member.

# ParentLabel Property

-

Returns a **ChCategoryLabel** object that represents the parent label for the specified label.

*expression*.**ParentLabel**

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

## Remarks

Using this property with a label that does not have a parent label will result in a run-time error.

# ParentMember Property

-  
Returns a [PivotMember](#) object that represents the parent member for the specified member.

*expression*.**ParentMember**

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

--

# ParentPageMember Property

-

Returns a **PivotPageMember** object that represents the parent member for the specified page member.

*expression*.**ParentPageMember**

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



# ParentRecordsetDef Property

-  
Returns a [RecordsetDef](#) object that represents the parent for the specified recordset definition.

*expression*.**ParentRecordsetDef**

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



# ParentRowMember Property

-

Returns a **PivotRowMember** object that represents the parent member for the specified row member.

*expression*.**ParentRowMember**

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

## Remarks

This property returns **Nothing** if this property is used for the top level row member.

--



# ParentSection Property

-  
Returns a [Section](#) object that represents the parent of the specified section.

*expression*.**ParentSection**

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

---

---

▼ [Show All](#)

# Path Property

-  
Returns a **String** that represents the path to the specified member. Read-only.

*expression*.**Path**(*Format*)

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

*Format* Required [PivotMemberFindFormatEnum](#).

PivotMemberFindFormatEnum can be one of these  
PivotMemberFindFormatEnum constants.

**plFindFormatMember** Not valid for this property.

**plFindFormatPathHex**

**plFindFormatPathInt**

**plFindFormatPathName**

---

▼ [Show All](#)

# Pattern Property

Returns a [ChartPatternTypeEnum](#) constant indicating the fill pattern for the specified **ChInterior** object. Read-only.

ChartPatternTypeEnum can be one of these ChartPatternTypeEnum constants.

**chPattern10Percent**

**chPattern20Percent**

**chPattern25Percent**

**chPattern30Percent**

**chPattern40Percent**

**chPattern50Percent**

**chPattern5Percent**

**chPattern60Percent**

**chPattern70Percent**

**chPattern75Percent**

**chPattern80Percent**

**chPattern90Percent**

**chPatternDarkDownwardDiagonal**

**chPatternDarkHorizontal**

**chPatternDarkUpwardDiagonal**

**chPatternDarkVertical**

**chPatternDashedDownwardDiagonal**

**chPatternDashedHorizontal**

**chPatternDashedUpwardDiagonal**

**chPatternDashedVertical**

**chPatternDiagonalBrick**

**chPatternDivot**

**chPatternDottedDiamond**

**chPatternDottedGrid**

**chPatternHorizontalBrick**  
**chPatternLargeCheckerBoard**  
**chPatternLargeConfetti**  
**chPatternLargeGrid**  
**chPatternLightDownwardDiagonal**  
**chPatternLightHorizontal**  
**chPatternLightUpwardDiagonal**  
**chPatternLightVertical**  
**chPatternNarrowHorizontal**  
**chPatternNarrowVertical**  
**chPatternOutlinedDiamond**  
**chPatternPlaid**  
**chPatternShingle**  
**chPatternSmallCheckerBoard**  
**chPatternSmallConfetti**  
**chPatternSmallGrid**  
**chPatternSolidDiamond**  
**chPatternSphere**  
**chPatternTrellis**  
**chPatternWave**  
**chPatternWeave**  
**chPatternWideDownwardDiagonal**  
**chPatternWideUpwardDiagonal**  
**chPatternZigZag**

*expression*.**Pattern**

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

## Remarks

Use the **SetPatterned** method to set the pattern for a **ChInterior** object.

# PercentComplete Property

-

Returns a **Long** value that represents the completed portion of the current operation. This property is supported only in the RecordsetSaveProgress event. Read-only.

*expression*.**PercentComplete**

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



## Remarks

Use this property with the **RecordsetSaveProgress** event to update a control in the container, such as the status bar.

Using this property with an unsupported event will result in a run-time error.

## Example

This example uses the RecordsetSaveProgress event to update the Microsoft Internet Explorer's status bar when the recordset contained by the **DataSourceControl** object is saved.

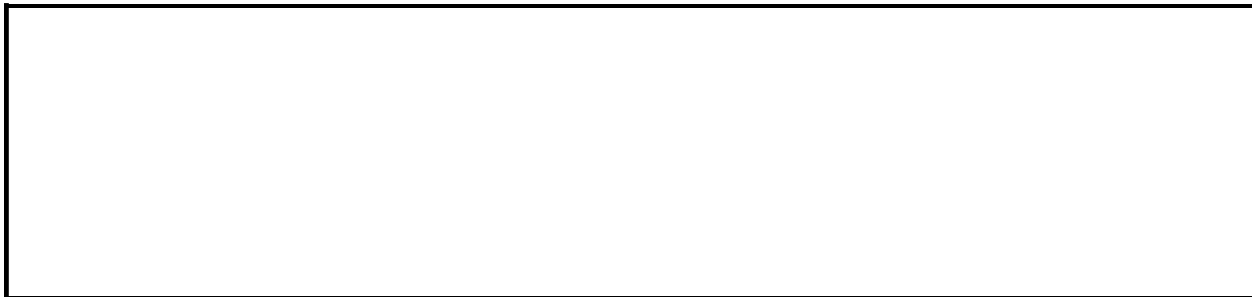
```
Sub MSODSC_RecordsetSaveProgress(DSCEventInfo)

    ' Update the status bar with the current
    ' completion percentage.
    Window.Status = DSCEventInfo.PercentComplete

    ' Check to see if the save has been completed.
    If DSCEventInfo.PercentComplete = 100 then

        ' Clear the status bar when the save is complete.
        Window.Status = ""
    End If

End Sub
```



# Period Property

-

Returns or sets a **Long** that represents the period for a moving-average trendline.  
Read/write.

*expression*.**Period**

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



# Perspective Property

-

Returns or sets a **Long** indicating the amount of perspective on a three-dimensional chart. This property has no effect if the **ProjectionMode** property of the chart has been set to **chProjectionModeOrthographic**. Valid values range from 0 to 80. Read/write.

*expression*.**Perspective**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then sets the perspective for the chart.

```
Sub SetPerspective()  
    Dim cht3DColumn  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Column chart.  
    cht3DColumn.Type = chChartTypeColumnClustered3D  
  
    ' Set the perspective.  
    cht3DColumn.Perspective = 35  
End Sub
```



# PivotAxis Property

Returns the **PivotResultGroupAxis** object for the specified category label or data series.

*expression*.**PivotAxis**

*expression* Required. An expression that returns a **ChCategoryLabels** or **ChSeriesCollection** object.



▼ [Show All](#)

# PivotObject Property

► [PivotObject property as it applies to the \*\*ChPoint\*\* object.](#)

Returns an **Object** that represents the corresponding PivotTable object for the specified point.

*expression*.**PivotObject**(*Dimension*)

*expression* Required. An expression that returns one of the above objects.

*Dimension* Required [ChartDimensionsEnum](#).

ChartDimensionsEnum can be one of these ChartDimensionsEnum constants.

**chDimBubbleValues**

**chDimCategories**

**chDimCharts**

**chDimCloseValues**

**chDimFilter**

**chDimFormatValues**

**chDimHighValues**

**chDimLowValues**

**chDimOpenValues**

**chDimRValues**

**chDimSeriesNames**

**chDimThetaValues**

**chDimValues**

**chDimXValues**

**chDimYValues**



► [PivotObject](#) property as it applies to the [ChCategoryLabel](#), [ChChartField](#), [ChLegendEntry](#), and [ChSeries](#) objects.

Returns an **Object** that represents the corresponding PivotTable object for the specified object.

*expression*.**PivotObject**

*expression* Required. An expression that returns one of the above objects.

## Remarks

This property returns **Null** if the chart is not bound to a relational data source.

▼ [Show All](#)

# PlotAllAggregates Property

-

Returns or sets a [ChartPlotAggregatesEnum](#) constant that determines which fields are plotted when the chart control is bound to a relational data source. Read/write.

ChartPlotAggregatesEnum can be one of these ChartPlotAggregatesEnum constants.

**chPlotAggregatesCategories** All data fields in the PivotTable list are plotted.

**chPlotAggregatesCharts**

**chPlotAggregatesFromTotalOrientation**

**chPlotAggregatesNone** Only the first data field in the PivotTable list is plotted.

**chPlotAggregatesSeries** Multiple data fields will be plotted as series nested at the innermost level.

*expression*.**PlotAllAggregates**

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

## Remarks

This property is relevant only when multiple fields have been added to the data area of the chart.

--

# PlotArea Property

-

Returns a [ChPlotArea](#) object that represents the plot area on the specified chart. Note that pie, doughnut, radar, and polar charts do not have plot areas.

*expression*.**PlotArea**

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

## Example

This example sets the interior color for the plot area on the specified chart.

```
ChartSpace1.Charts(0).PlotArea.Interior.Color = "Green"
```



# Points Property

-  
Returns the [ChPoints](#) collection for the specified series.

*expression*.**Points**

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



## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

## Example

This example sets the interior color for point two in the specified series.

```
ChartSpace1.Charts(0).SeriesCollection(0).Points(2).Interior.Color =
```



▼ [Show All](#)

# Position Property

Returns or sets the position of the object.

- Read/write [ChartAxisPositionEnum](#) for the [ChAxis](#) object.

ChartAxisPositionEnum can be one of these ChartAxisPositionEnum constants.

**chAxisPositionBottom**

**chAxisPositionCategory**

**chAxisPositionCircular**

**chAxisPositionLeft**

**chAxisPositionPrimary**

**chAxisPositionRadial**

**chAxisPositionRight**

**chAxisPositionSecondary**

**chAxisPositionSeries**

**chAxisPositionTimescale**

**chAxisPositionTop**

**chAxisPositionValue**

- Read/write [ChartDataLabelPositionEnum](#) for the [ChDataLabels](#) object.

ChartDataLabelPositionEnum can be one of these  
ChartDataLabelPositionEnum constants.

**chLabelPositionAutomatic**

**chLabelPositionBottom**

**chLabelPositionCenter**

**chLabelPositionInsideBase**

**chLabelPositionInsideEnd**

**chLabelPositionLeft**

**chLabelPositionOutsideBase**

**chLabelPositionOutsideEnd**  
**chLabelPositionRight**  
**chLabelPositionTop**

- Read/write [ChartLegendPositionEnum](#) for the [ChLegend](#) object.

ChartLegendPositionEnum can be one of these ChartLegendPositionEnum constants.

**chLegendPositionAutomatic**  
**chLegendPositionBottom**  
**chLegendPositionLeft**  
**chLegendPositionRight**  
**chLegendPositionTop**

- Read/write [ChartTitlePositionEnum](#) for the [ChTitle](#) object.

ChartTitlePositionEnum can be one of these ChartTitlePositionEnum constants.

**chTitlePositionAutomatic**  
**chTitlePositionBottom**  
**chTitlePositionLeft**  
**chTitlePositionRight**  
**chTitlePositionTop**

*expression*.**Position**

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

## Example

This example puts the legend for the specified chart to the left of the plot area.

```
Sub AddLegend()  
    Dim chConstants  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Enable the legend for the first chart in Chartspace1.  
    ChartSpace1.Charts(0).HasLegend = True  
  
    ' Set the position of the legend.  
    ChartSpace1.Charts(0).Legend.Position = chConstants.chLegendPosit  
End Sub
```



# Precision Property

-  
Returns the maximum number of digits you can use for numeric types. Read-only **Long**.

*expression*.**Precision**

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

--

# PrefixCharacter Property

-

Returns the prefix character for the specified range. Returns an apostrophe (') if the specified range contains a value that was preceded with an apostrophe when it was entered into the worksheet. Returns a blank string if the specified range does not contain a prefix character. Read-only **Variant**.

*expression*.**PrefixCharacter**

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



## Example

This example deletes all prefix characters from the active worksheet in Spreadsheet1.

```
Sub Delete_PrefixCharacters()  
    Dim rngCell  
  
    ' Loop through all used cells in the active worksheet.  
    For Each rngCell In Spreadsheet1.ActiveSheet.UsedRange  
  
        ' If a prefix character exists, delete it.  
        If rngCell.PrefixCharacter <> "" Then  
            rngCell.Value = rngCell.Value  
        End If  
    Next  
End Sub
```



▼ [Show All](#)

# PresetGradientType Property

Returns a [ChartPresetGradientTypeEnum](#) constant indicating the preset gradient type for the specified **ChInterior** object. Read-only.

ChartPresetGradientTypeEnum can be one of these ChartPresetGradientTypeEnum constants.

**chGradientBrass**

**chGradientCalmWater**

**chGradientChrome**

**chGradientChromeII**

**chGradientDaybreak**

**chGradientDesert**

**chGradientEarlySunset**

**chGradientFire**

**chGradientFog**

**chGradientGold**

**chGradientGoldII**

**chGradientHorizon**

**chGradientLateSunset**

**chGradientMahogany**

**chGradientMoss**

**chGradientNightfall**

**chGradientOcean**

**chGradientParchment**

**chGradientPeacock**

**chGradientRainbow**

**chGradientRainbowII**

**chGradientSapphire**

**chGradientSilver**

## **chGradientWheat**

*expression*.**PresetGradientType**

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

## Remarks

Use the **SetPresetGradient** method to set the preset gradient type for the fill.

▼ [Show All](#)

# PresetTexture Property

Returns a [ChartPresetTextureEnum](#) constant indicating the preset texture for the specified **ChInterior** object. Read-only.

ChartPresetTextureEnum can be one of these ChartPresetTextureEnum constants.

**chTextureBlueTissuePaper**

**chTextureBouquet**

**chTextureBrownMarble**

**chTextureCanvas**

**chTextureCork**

**chTextureDenim**

**chTextureFishFossil**

**chTextureGranite**

**chTextureGreenMarble**

**chTextureMediumWood**

**chTextureNewsprint**

**chTextureOak**

**chTexturePaperBag**

**chTexturePapyrus**

**chTextureParchment**

**chTexturePinkTissuePaper**

**chTexturePurpleMesh**

**chTextureRecycledPaper**

**chTextureSand**

**chTextureStationery**

**chTextureWalnut**

**chTextureWaterDroplets**

**chTextureWhiteMarble**

**chTextureWovenMat**

*expression*.**PresetTexture**

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



## Remarks

Use the **SetTextured** method to set the preset texture for a **ChInterior** object.

▼ [Show All](#)

# Previous Property

## ▶ [Previous property as it applies to the \*\*Range\*\* object.](#)

Returns a **Range** object that represents the previous cell. 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.

Using this property when the active cell in the specified range is in column A will result in a run-time error.

*expression*.**Previous**

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

## ▶ [Previous property as it applies to the \*\*Worksheet\*\* object.](#)

Returns a **Worksheet** object that represents the previous sheet.

*expression*.**Previous**

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

## Example

► [As it applies to the \*\*Range\*\* object.](#)

This example selects the previous unlocked cell on the active worksheet.

```
Spreadsheet1.ActiveCell.Previous.Select
```



# PreviousSection Property

-

Returns a [Section](#) object that represents the previous section on the specified page, regardless of siblings or parents. This property fails on the first section of the page.

*expression*.**PreviousSection**

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

---

---

# PreviousSibling Property

-

Returns a [Section](#) object that represents the previous sibling in the current data access page. This property fails on the first sibling of the current data access page.

*expression*.**PreviousSibling**

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

---

---

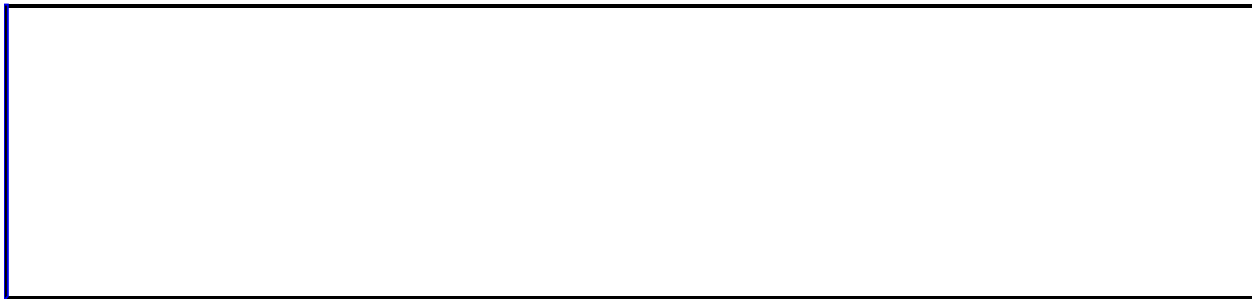
# PrimaryPageRowsource Property

-

Returns a [PageRowsource](#) object that represents the primary page row source for the specified recordset definition. When a recordset definition is created, the row source named in the **Add** or **AddNew** method becomes the primary page row source.

*expression*.**PrimaryPageRowsource**

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



# PrintQuality3D Property

-

Returns or sets a **Double** indicating the ratio of the printed resolution of a three-dimensional (3-D) chart to the resolution of your printer. Valid values range from 0 to 1. The default value is 0.25. Read/write.

*expression*.**PrintQuality3D**

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



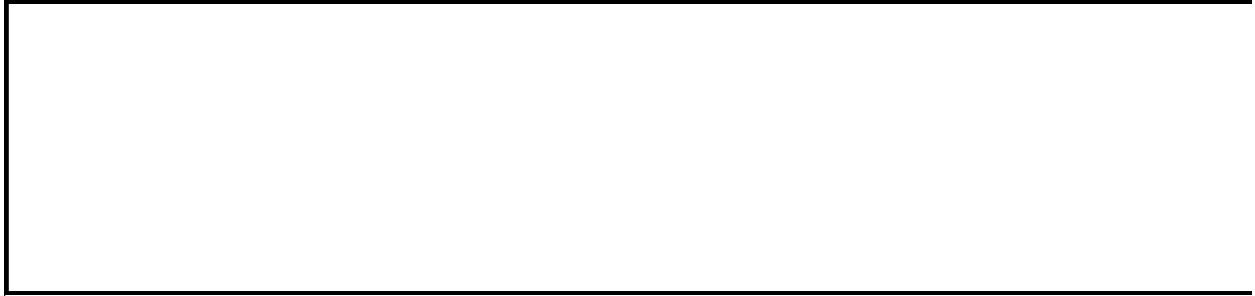
## Remarks

The lower the value of this property, the faster your 3-D chart will print.  
However, setting this value of this property too low may yield unsatisfactory results.

## Example

This example sets the 3-D charts in Chartspace1 to print at 75% of the printer's resolution.

```
Chartspace1.PrintQuality3D = 0.75
```



▼ [Show All](#)

# ProjectionMode Property

Returns or sets a [ChartProjectionModeEnum](#) constant indicating the viewing perspective of a three-dimensional chart. Read/write.

ChartProjectionModeEnum can be one of these ChartProjectionModeEnum constants.

**chProjectionModeOrthograph** Perspective is not applied in this view. The advantage to using this type of projection is that the vertical lines remain vertical, making some charts easier to read.

**chProjectionModePerspective** Provides the most realistic 3-D appearance. Objects farther away converge towards a vanishing point. This is the default projection.

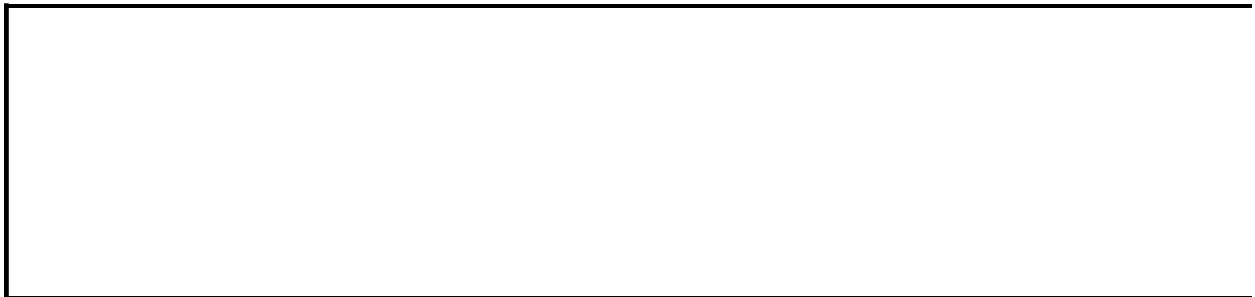
*expression*.**ProjectionMode**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then sets the projection mode of the chart.

```
Sub SetExtrudeAngle()  
  
    Dim cht3DColumn  
    Dim chConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Column chart.  
    cht3DColumn.Type = chChartTypeColumnClustered3D  
  
    ' Sets the projection mode to orthographic.  
    cht3DColumn.ProjectionMode = chConstants.chProjectionModeOrthogr  
  
    ' Sets the extrusion angle.  
    cht3DColumn.ExtrudeAngle = 75  
  
End Sub
```



# PropertyCaptionFont Property

-

Returns a **PivotFont** object that represents the font settings used to display the caption of OLAP member properties in the specified view. Use the returned object to format the font used for member property captions.

*expression*.**PropertyCaptionFont**

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

## Example

This example formats the alignment and the font of member property captions and values in the active view of PivotTable1.

```
Sub Format_MemberProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' The following three lines of code format the  
    ' font for member property captions.  
    ptView.PropertyCaptionFont.Name = "Tahoma"  
    ptView.PropertyCaptionFont.Size = 8  
    ptview.PropertyCaptionFont.Bold = True  
  
    ' Left-align the member property captions.  
    ptview.PropertyCaptionHAlignment = ptConstants.plHAlignLeft  
  
    ' The following two lines of code format the  
    ' font for member property values.  
    ptview.PropertyValueFont.Name = "Tahoma"  
    ptview.PropertyValueFont.Size = 8  
  
    ' Right-align the member property values.  
    ptview.PropertyValueHAlignment = ptConstants.plHAlignRight  
  
End Sub
```



▼ [Show All](#)



# PropertyCaptionHAlignment Property

-

Returns or sets a [PivotHAlignmentEnum](#) constant that represents the horizontal alignment of OLAP member property captions in the specified view. Use this property to set the horizontal alignment of member property captions.  
Read/write.

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

*expression*.**PropertyCaptionHAlignment**

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

## Example

This example formats the alignment and the font of member property captions and values in the active view of PivotTable1.

```
Sub Format_MemberProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' The following three lines of code format the  
    ' font for member property captions.  
    ptView.PropertyCaptionFont.Name = "Tahoma"  
    ptView.PropertyCaptionFont.Size = 8  
    ptview.PropertyCaptionFont.Bold = True  
  
    ' Left-align the member property captions.  
    ptview.PropertyCaptionHAlignment = ptConstants.plHAlignLeft  
  
    ' The following two lines of code format the  
    ' font for member property values.  
    ptview.PropertyValueFont.Name = "Tahoma"  
    ptview.PropertyValueFont.Size = 8  
  
    ' Right-align the member property values.  
    ptview.PropertyValueHAlignment = ptConstants.plHAlignRight  
  
End Sub
```



# PropertyCaptionWidth Property

-

Returns or sets a **Long** that represents the width of the caption of a member property that is displayed in the specified field. Read/write.

*expression*.**PropertyCaptionWidth**

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



# PropertyHeight Property

-

Returns or sets a **Long** that represents the height of a member property that is displayed in the specified field. Read/write.

*expression*.**PropertyHeight**

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



# PropertyValueFont Property

-

Returns a **PivotFont** object that represents the font settings used to display the value of OLAP member properties in the specified view. Use the returned object to format the font used for member property values.

*expression*.**PropertyValueFont**

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

## Example

This example formats the alignment and the font of member property captions and values in the active view of PivotTable1.

```
Sub Format_MemberProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' The following three lines of code format the  
    ' font for member property captions.  
    ptView.PropertyCaptionFont.Name = "Tahoma"  
    ptView.PropertyCaptionFont.Size = 8  
    ptview.PropertyCaptionFont.Bold = True  
  
    ' Left-align the member property captions.  
    ptview.PropertyCaptionHAlignment = ptConstants.plHAlignLeft  
  
    ' The following two lines of code format the  
    ' font for member property values.  
    ptview.PropertyValueFont.Name = "Tahoma"  
    ptview.PropertyValueFont.Size = 8  
  
    ' Right-align the member property values.  
    ptview.PropertyValueHAlignment = ptConstants.plHAlignRight  
  
End Sub
```



▼ [Show All](#)

# PropertyValueHAlignment Property

-

Returns or sets a [PivotHAlignmentEnum](#) constant that represents the horizontal alignment of OLAP member property values in the specified view. Use this property to set the horizontal alignment of member property values. Read/write.

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

*expression*.**PropertyValueHAlignment**

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



## Example

This example formats the alignment and the font of member property captions and values in the active view of PivotTable1.

```
Sub Format_MemberProperties()  
  
    Dim ptView  
    Dim ptConstants  
  
    Set ptConstants = PivotTable1.Constants  
  
    ' Set a variable to the current PivotTable view.  
    Set ptView = PivotTable1.ActiveView  
  
    ' The following three lines of code format the  
    ' font for member property captions.  
    ptView.PropertyCaptionFont.Name = "Tahoma"  
    ptView.PropertyCaptionFont.Size = 8  
    ptview.PropertyCaptionFont.Bold = True  
  
    ' Left-align the member property captions.  
    ptview.PropertyCaptionHAlignment = ptConstants.plHAlignLeft  
  
    ' The following two lines of code format the  
    ' font for member property values.  
    ptview.PropertyValueFont.Name = "Tahoma"  
    ptview.PropertyValueFont.Size = 8  
  
    ' Right-align the member property values.  
    ptview.PropertyValueHAlignment = ptConstants.plHAlignRight  
  
End Sub
```



# PropertyValueWidth Property

-

Returns or sets a **Long** that represents the width of the value of a member property that is displayed in the specified field. Read/write.

*expression*.**PropertyValueWidth**

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

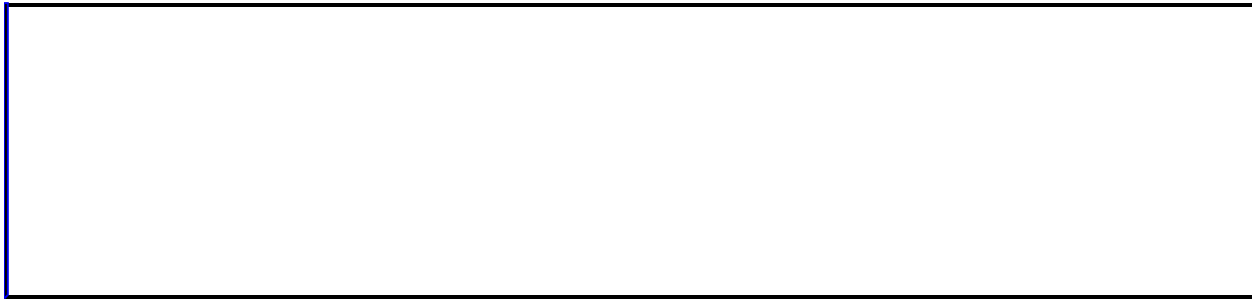


# ProtectContents Property

**True** if the contents of the sheet are protected. Read-only **Boolean**.

*expression*.**ProtectContents**

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



# Protection Property

-  
Returns a [Protection](#) object that contains the protection properties for the specified worksheet. The protection properties are ignored if the [Enabled](#) property is set to **False**.

*expression*.**Protection**

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

## Example

This example locks the cells in column B and enables protection for the sheet.

```
Sub ProtectColumnB()  
    ' Unlock all of the cells in the active sheet.  
    Spreadsheet1.ActiveSheet.Cells.Locked = False  
  
    ' Lock the cells in column B.  
    Spreadsheet1.Columns(2).Locked = True  
  
    ' Protect the locked cells.  
    Spreadsheet1.ActiveSheet.Protection.Enabled = True  
End Sub
```



# ProtectionMode Property

**True** if user-interface-only protection is enabled. Always returns **True**. Read-only **Boolean**.

*expression*.**ProtectionMode**

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



# ProtectStructure Property

**True** if the order of the sheets in the workbook is protected. Read-only **Boolean**.

*expression*.**ProtectStructure**

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

---

---

# ProviderFormattedValue Property

-

Returns a **Variant** that represents the value of the specified aggregate as it is formatted by the data source. Read-only.

*expression*.**ProviderFormattedValue**

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





# ProviderType Property

-

Returns a **ProviderType** constant that represents the type of data provider for the specified PivotTable list. Read-only.

*expression*.**ProviderType**

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



▼ [Show All](#)

# Range Property

## ▶ [Range property as it applies to the \*\*AutoFilter\*\* object.](#)

Returns a **Range** object that represent the range of cells that the specified **AutoFilter** object applies to.

*expression*.**Range**

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

## ▶ [Range property as it applies to the \*\*PivotData\*\* object.](#)

Returns a **PivotRange** object that represents a cell or a range of cells.

*expression*.**Range**(*TopLeft*, *BottomRight*)

*expression* An expression that returns a **PivotData** object.

**TopLeft** Required [PivotCell](#) object. Specifies the upper-left cell in the specified range.

**BottomRight** Required [PivotCell](#) object. Specifies the lower-right cell in the specified range.

## ▶ [Range property as it applies to the \*\*Range\*\*, \*\*Spreadsheet\*\*, and \*\*Worksheet\*\* objects.](#)

Returns a **Range** object that represents a cell or rectangular range of cells. It is not possible to create a nonrectangular range or a range composed of discontinuous areas.

*expression*.**Range**(*Cell1*, *Cell2*)

*expression* An expression that returns a **Range**, **Spreadsheet**, or **Worksheet** object.

**Cell1** Required **Variant**. Specifies the entire range as an A1-style reference ("A1:C3", for example). The reference 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 these are ignored. The reference can also specify the cell in the upper-left corner of the range as a **Range** object that contains a single cell, an entire column, or an entire row, or as a string that names a single cell. If **Cell1** specifies the upper-left cell in the range, **Cell2** is required.

**Cell2** Optional **Variant**. (Required if **Cell1** specifies the upper-left cell in the specified range.) Specifies the cell in the lower-right corner of the range. Can be a **Range** object that contains a single cell, an entire column, or an entire row, or it can be a string that names a single cell.

## Remarks

When this property is applied to a **Range** object, the arguments specify cells relative to the range. The following example sets a variable to a **Range** object that represents cell B5.

```
Set tempRange = Spreadsheet1.Range("b5:b10").range("a1")
```

## Example

This example sets the row height to 15 points for rows 1 through 10.

```
Spreadsheet1.Range("a1:a10").RowHeight = 15
```

This example sets a variable to a range that includes the second through last row of columns 3 and 4 in the current region for cell A1.

```
Sub GetRange()
```

```
    Dim rngCurRegion  
    Dim rngTempRange
```

```
    Set rngCurRegion = Spreadsheet1.Range("a1").CurrentRegion
```

```
    Set rngTempRange = rngCurRegion.Range(rngCurRegion.Cells(2, 3), _  
        curRegion.Cells(rngCurRegion.Rows.Count, 4))
```

```
End Sub
```



# RangeSelection Property

-

Returns a **Range** object that represents the selected cells on the worksheet in the specified window. Read-only.

*expression*.**RangeSelection**

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

## Remarks

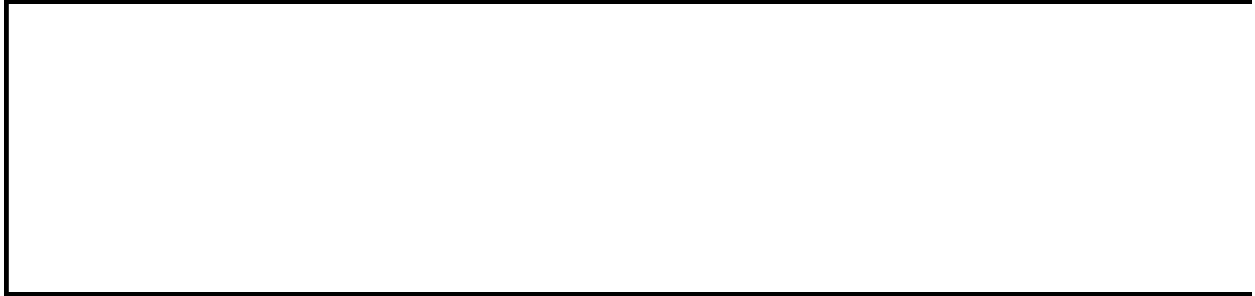
This property is identical in functionality to the **Selection** property.



## Example

This example sets sets the font of the selected cells to bold.

```
Spreadsheet1.ActiveWindow.RangeSelection.Font.Bold = True
```



▼ [Show All](#)

# ReadingOrder Property

-  
Returns or sets a **Long** representing the reading order for the specified object.  
Can be a [XLReadingOrder](#) constant. Read/write.

XlLineStyle can be one of these XlLineStyle constants.

**xlRTL** Right-to-left

**xlLTR** Left-to-right

**xlContext**

*expression*.**ReadingOrder**

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

## Remarks

Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

# RecordNavigationSection Property

**True** if there is a navigation section for any given banding section. The default value is **True**. Read/write **Boolean**.

*expression*.**RecordNavigationSection**

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

--

# RecordSelector Property

-

Returns or sets a **Boolean** that indicates whether or not to display the record selector for the specified group. Read/write.

*expression*.**RecordSelector**

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

--

# Recordset Property

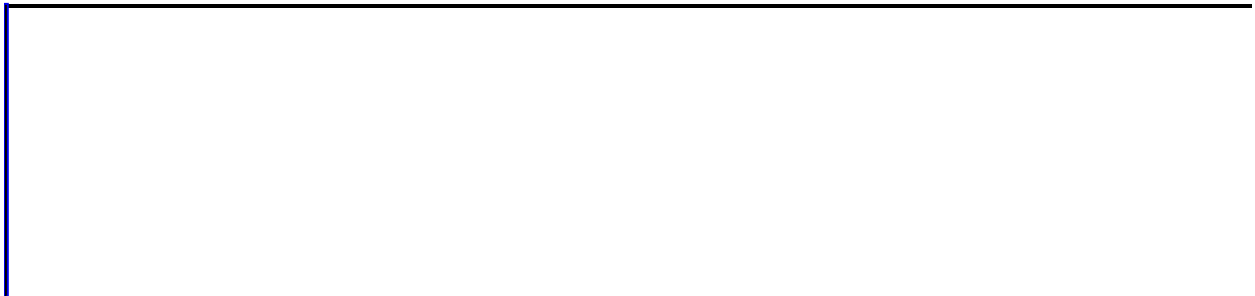
-  
**PivotCell** object: Returns an ADO **Recordset** object that contains the detail records for the cell if they are available.

**PivotData** object: Returns an ADO **Recordset** object that contains the detail records for the specified **PivotData** object if they are available.

**DataPage** object: Returns an ADO **Recordset** object for the specified data access page.

*expression*.**Recordset**

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



# RecordsetDef Property

-  
Returns the containing [RecordsetDef](#) object for the specified page field or page row source.

*expression*.**RecordsetDef**

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

--



# RecordsetDefs Property

-  
Returns the [RecordsetDefs](#) collection for the data source control.

*expression*.**RecordsetDefs**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# RecordsetLabel Property

-

Returns or sets the recordset label for the record navigation control when the page is not banded and one record is displayed, and when the page is banded and two or more records are displayed. Read/write **String**.

*expression*.**RecordsetLabel**

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

## Remarks

The label consists of two strings separated by a semicolon. The first string represents the label that is displayed when one record is displayed on the page, and the second string represents the label that is displayed when two or more records are displayed on the page. The label can contain the following placeholders for displaying recordset information on the page.

Placeholder	Description
0	The number of the current record, or the number of the first visible record in the group.
1	The number of the last visible record in the group.
2	The number of records in the recordset.

## Example

This example sets the recordset label. If the page is banded and records 1 through 5 out of 8 are displayed, the label says "Categories 1-5 of 8." If the page is not banded and the first record is displayed, the label says "Category 1 of 8."

```
ProductNavigation.RecordsetLabel = _  
    "Category |1 of |2;Categories |0-|1 of |2"
```



▼ [Show All](#)

# RecordsetType Property

-

Returns or sets the recordset type for the data source control. Read/write [DscRecordsetTypeEnum](#).

DscRecordsetTypeEnum can be one of these DscRecordsetTypeEnum constants.

**dscSnapshot**

**dscUpdatableSnapshot**

*expression*.**RecordsetType**

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

---

---

# RecordSource Property

-

**ElementExtension** and **GroupLevel** objects: Returns or sets a **String** that represents the record source (the name of a recordset definition or grouping definition) for the section. Applies only to DIV sections. Read/write.

**RecordNavigationControl** object: Returns or sets a **DataMember** object that represents the record source for the section. Read/write.

*expression*.**RecordSource**

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

--



# RefersTo Property

-

Returns or sets the formula that the name as defined refers to, in the language of the script and in A1-style notation, beginning with an equal sign. Read/write **Variant**.

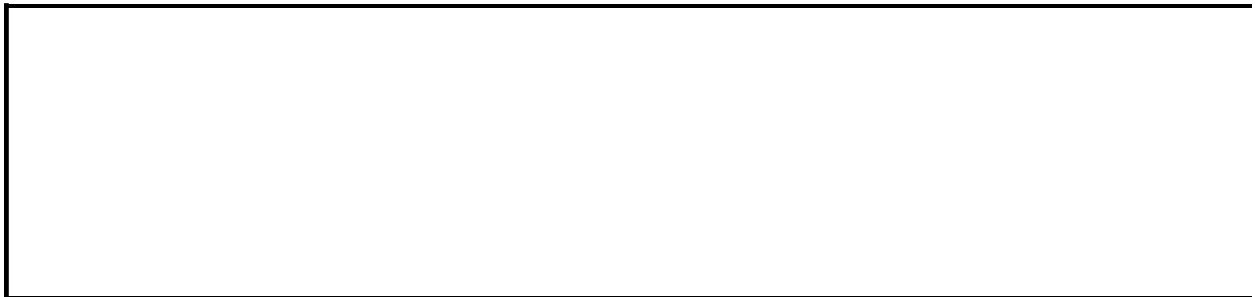
*expression*.**RefersTo**

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

## Example

The following example creates a list of all the names in the active workbook, along with the formulas to which they refer.

```
Sub List_All_Names()  
    Dim nmCurrentName  
    Dim rngCurrent  
  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Range("A1")  
  
    ' Loop through all of the names in the active workbook.  
    For Each nmCurrentName In Spreadsheet1.ActiveWorkbook.Names  
  
        ' Write the current name to the worksheet.  
        rngCurrent.Value = nmCurrentName.Name  
  
        ' Write the definition of the current name to the worksheet.  
        rngCurrent.Offset(0, 1).Value = "" & nmCurrentName.RefersTo  
  
        Set rngCurrent = rngCurrent.Offset(1, 0)  
    Next  
End Sub
```



# RefersToLocal Property

-

Returns or sets a **Variant** representing 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 equals sign. Read/write.

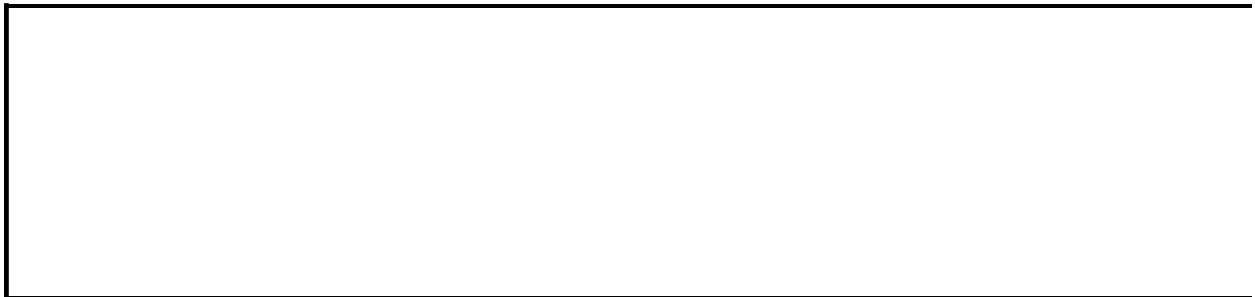
*expression*.**RefersToLocal**

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

## Example

The following example creates a list of all the names in the active workbook, along with the formulas to which they refer, in the language of the user.

```
Sub List_All_Names()  
    Dim nmCurrentName  
    Dim rngCurrent  
  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Range("A1")  
  
    ' Loop through all of the names in the active workbook.  
    For Each nmCurrentName In Spreadsheet1.ActiveWorkbook.Names  
  
        ' Write the current name to the worksheet.  
        rngCurrent.Value = nmCurrentName.Name  
  
        ' Write the definition of the current name to the worksheet.  
        rngCurrent.Offset(0, 1).Value = "'" & nmCurrentName.ReferentsToLo  
  
        Set rngCurrent = rngCurrent.Offset(1, 0)  
    Next  
End Sub
```



# RefersToRange Property

-

Returns the **Range** object referred to by a **Name** object. If the **Name** object doesn't refer to a range (for example, if it refers to a constant or a formula), this property generates a run-time error. Read-only.

*expression*.**RefersToRange**

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

---

---

# ResyncCommand Property

-  
Specifies an SQL command parameterized by the key field values from the specified recordset's unique table such that the command returns exactly one record. The resync command is executed to "fix up" a row after an update or insertion is made. Read/write **String**.

*expression*.**ResyncCommand**

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

## Remarks

You must set this property for any form based on a stored procedure that contains a join or expression column. In all other cases, Microsoft Access can fix tables automatically. If you do not set this property, users will not see fixed-up field values after an update or insertion, but the update or insertion will still be executed correctly.

# ReturnValue Property

-

Returns or sets the return value for the specified event. You can cancel the default action for some events by setting this property to **False**. Read/write **Boolean**.

*expression*.**ReturnValue**

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



## Example

This example uses the **BeforeInsert** event to prevent the user from adding another record to the recordset once it reaches 75 records.

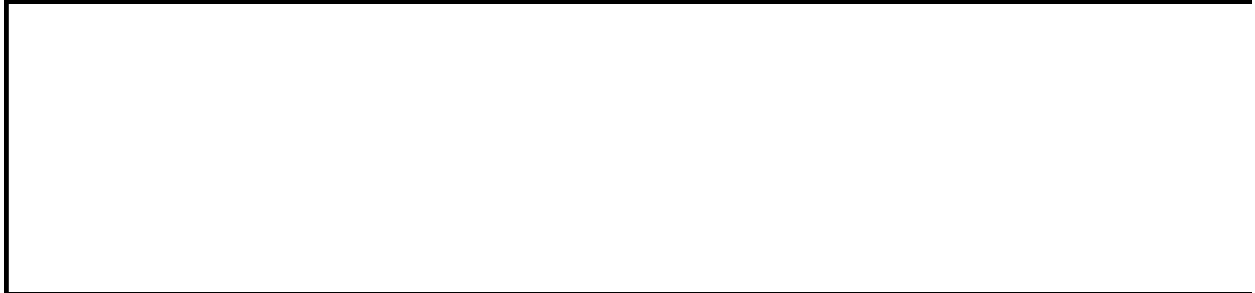
```
Sub MSODSC_BeforeInsert(DSCEventInfo)
    Dim rstCurrentData

    ' Set a variable to the recordset.
    Set rstCurrentData = DSCEventInfo.DataPage.Recordset

    ' Check to see if the recordset has reached its limit.
    If rstCurrentData.RecordCount >= 75 then

        ' Display a message to the user.
        MsgBox "Cannot add any more records."

        ' Cancel the insertion of the record.
        DSCEventInfo.ReturnValue = False
    End If
End Sub
```



# RevisionNumber Property

-

Returns the Microsoft Office Web Components revision number. Read-only **String**.

*expression*.**RevisionNumber**

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

--

▼ [Show All](#)

# Right Property

▸ [Right property as it applies to the \*\*ChPlotArea\*\* object.](#)

Returns or sets a **Long** that represents the right edge of the specified object.  
Read/write.

*expression*.**Right**

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

▸ [Right property as it applies to the \*\*ChartSpace\*\*, \*\*ChAxis\*\*, \*\*ChCategoryLabel\*\*, \*\*ChChart\*\*, \*\*ChChartField\*\*, \*\*ChDataLabel\*\*, \*\*ChDataLabels\*\*, \*\*ChDropZone\*\*, \*\*ChErrorBars\*\*, \*\*ChLegend\*\*, \*\*ChLegendEntry\*\*, \*\*ChPoint\*\*, \*\*ChSeries\*\*, \*\*ChTitle\*\*, and \*\*ChTrendline\*\* objects.](#)

Returns a **Long** that represents the right edge of the specified object. Read-only.

*expression*.**Right**

*expression* Required. An expression that returns one of the above objects.



# RightToLeft Property

-  
**True** if right-to-left language support is enabled. For example, setting this property to **True** causes scroll bars to be displayed on the left. Read/write **Boolean**.

*expression*.**RightToLeft**

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

--

# Rotation Property

-

Returns or sets a **Double** indicating the rotation in degrees of the specified three-dimensional chart. Valid values range from 0 to 360. Read/write.

*expression*.**Rotation**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then rotates the chart 145 degrees.

```
Sub SetGapDepth()  
    Dim cht3DColumn As ChChart  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3-D Column chart.  
    cht3DColumn.Type = chChartTypeColumn3D  
  
    ' Rotate the chart.  
    cht3DColumn.Rotation = 145  
End Sub
```



# Row Property

-  
Returns the number of the first row in the specified range. Read-only **Long**.

*expression*.**Row**

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



## Example

This example makes every other row green in the current region for cell A1.

```
Sub FormatRows()  
    Dim rngCurrentRow  
  
    ' Loop through all of the rows in the current region.  
    For Each rngCurrentRow In Spreadsheet1.Cells(1, 1).CurrentRegion.  
        ' Check to see if the row number is an even number.  
        If rngCurrentRow.Row Mod 2 = 0 Then  
            ' Set the interior color of the row.  
            rngCurrentRow.Interior.Color = "LightGreen"  
        End If  
    Next  
End Sub
```



# RowAxis Property

-  
**PivotData** object: Returns a [PivotResultRowAxis](#) object that represents the row axis.

**PivotView** object: Returns a [PivotGroupAxis](#) object that represents the row axis.

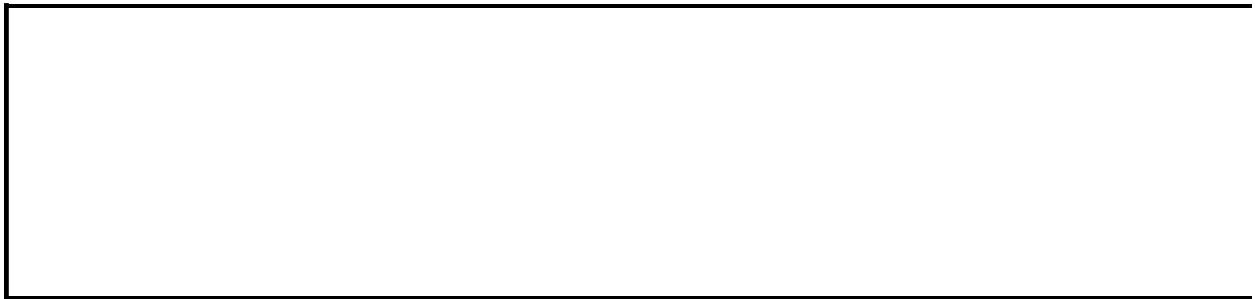
*expression*.**RowAxis**

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

## Example

This example inserts two field sets into the PivotTable list in the active view.

```
Sub AddFieldsToPT()  
    Dim ptView  
    Set ptView = PivotTable1.ActiveView  
    ptView.ColumnAxis.InsertFieldSet ptView.FieldSets("Store Type")  
    ptView.RowAxis.InsertFieldSet ptView.FieldSets("Promotions")  
End Sub
```



# RowHeadings Property

-

Returns a **Headings** collection that represents the row headings in the specified window. Use the **Caption** property to customize the row headings.

*expression*.**RowHeadings**

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

## Example

This example sets the creates a custom data entry sheet by disabling some user interface elements, limiting the viewable range in the active window, and customizing the row and column headings.

```
Sub Create_Datasheet()  
    Dim hdrColHeadings  
    Dim hdrRowHeadings  
    Dim wndActive  
  
    Set wndActive = Spreadsheet1.ActiveWindow  
  
    ' Hide various UI elements.  
    wndActive.DisplayWorkbookTabs = False  
    Spreadsheet1.DisplayToolbar = False  
    Spreadsheet1.DisplayScrollBars = False  
  
    ' Display the title bar and set it's caption.  
    Spreadsheet1.DisplayTitleBar = True  
    Spreadsheet1.TitleBar.Caption = "Revenue Worksheet"  
  
    ' Resize the spreadsheet component.  
    Spreadsheet1.AutoFit = True  
  
    ' Limit the viewable range of the active sheet.  
    wndActive.ViewableRange = "A1:D5"  
  
    ' Set a variable to the column headings in the active window.  
    Set hdrColHeadings = wndActive.ColumnHeadings  
  
    ' Set a variable to the row headings in the active window.  
    Set hdrRowHeadings = wndActive.RowHeadings  
  
    ' Set the headings of columns A through D.  
    hdrColHeadings(1).Caption = "Qtr 1"  
    hdrColHeadings(2).Caption = "Qtr 2"  
    hdrColHeadings(3).Caption = "Qtr 3"  
    hdrColHeadings(4).Caption = "Qtr 4"  
  
    ' Set the headings of rows 1 though 5.  
    hdrRowHeadings(1).Caption = "1996"  
    hdrRowHeadings(2).Caption = "1997"  
    hdrRowHeadings(3).Caption = "1998"  
    hdrRowHeadings(4).Caption = "1999"  
    hdrRowHeadings(5).Caption = "2000"
```

End Sub

--

# RowHeight Property

-

Returns or sets the height (in points) of all rows in the specified range. Returns **Null** if the rows are not all the same height. Use the **IsNull** function to determine whether the return value is **Null**. Read/write **Variant**.

*expression*.**RowHeight**

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

## Example

This example sets the row height to 15 points for rows 1 through 10.

```
Spreadsheet1.Range("a1:a10").RowHeight = 15
```





# RowMember Property

-

Returns a [PivotRowMember](#) object that represents the inner member on the row axis that intersects the specified cell.

*expression*.**RowMember**

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

--

# RowMembers Property

-  
Returns a [PivotRowMembers](#) collection that contains the members on the row axis.

*expression*.**RowMembers**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# Rows Property

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

**Spreadsheet** object: Returns a **Range** object that represents all the rows on the active worksheet.

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

*expression*.**Rows**

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

## Example

This example makes every other row bold in the current region for cell A1.

```
Sub BoldEvenRows()  
    Dim rngCurrentRow  
    ' Loop through the rows in the current region.  
    For Each rngCurrentRow In Spreadsheet1.Cells(1, 1).CurrentRegion.  
        ' Check whether the current row number is an even number.  
        If rngCurrentRow.Row Mod 2 = 0 Then  
            ' Bold the font in the current row.  
            rngCurrentRow.Font.Bold = True  
        End If  
    Next  
End Sub
```

This example makes row 2 bold.

```
Spreadsheet1.Rows(2).Font.Bold = True
```



# Scaling Property

-  
Returns the **ChScaling** object for the specified axis.

*expression*.**Scaling**

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

## Example

This example adds a second value (y) axis to the right side of the specified chart. The second axis uses the same scale as the primary value axis.

```
Sub AddAxis()  
    Dim chConstants  
    Dim axisScale  
  
    Set chConstants = ChartSpace1.Constants  
    Set axisScale = ChartSpace1.Charts(0).Axes(chConstants.chAxisPosi  
ChartSpace1.Charts(0).Axes.Add axisScale, chConstants.chAxisPosit  
                                chConstants.chValueAxis  
End Sub
```



▼ [Show All](#)



# Scalings Property

Returns a [ChScaling](#) object for the specified chart or series.

*expression*.**Scalings**(*Dimension*)

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

***Dimension*** Required [ChartDimensionsEnum](#). Specifies the dimension to be returned.

ChartDimensionsEnum can be one of these ChartDimensionsEnum constants.

**chDimBubbleValues**

**chDimCategories**

**chDimCharts**

**chDimCloseValues**

**chDimFilter**

**chDimFormatValues**

**chDimHighValues**

**chDimLowValues**

**chDimOpenValues**

**chDimRValues**

**chDimSeriesNames**

**chDimThetaValues**

**chDimValues**

**chDimXValues**

**chDimYValues**

## Example

This example sets the minimum value for the specified **ChScaling** object.

```
Sub SetScaling()  
    Dim chConstants  
    Set chConstants = ChartSpace1.Constants  
    ChartSpace1.Charts(0).SeriesCollection(0).Scalings _  
        (chConstants.chDimCategories).Minimum = -10  
End Sub
```



# SchemaFields Property

-  
Returns the [SchemaFields](#) collection for the specified schema row source.

*expression*.**SchemaFields**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# SchemaParameters Property

-  
Returns the [SchemaParameters](#) collection for the specified schema row source.

*expression*.**SchemaParameters**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# SchemaRelatedFields Property

-  
Returns the [SchemaRelatedFields](#) collection for the specified schema relationship.

*expression*.**SchemaRelatedFields**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

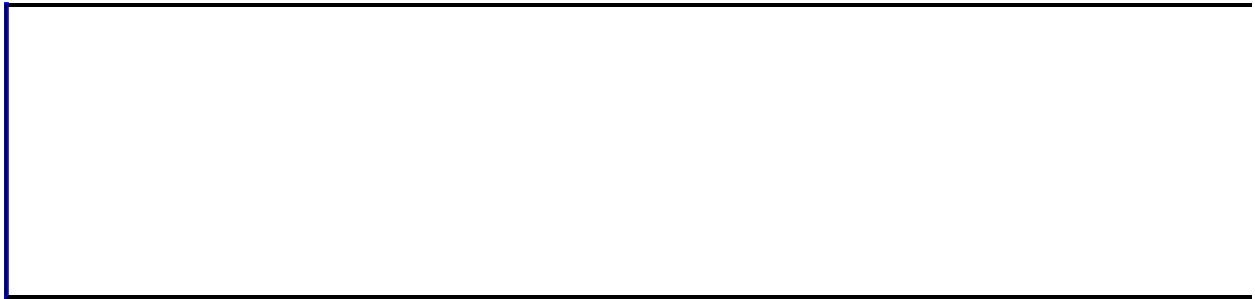


# SchemaRelationships Property

-  
Returns the [SchemaRelationships](#) collection for the data source control.

*expression*.**SchemaRelationships**

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



# SchemaRowsources Property

-  
Returns the [SchemaRowsources](#) collection for the data source control.

*expression*.**SchemaRowsources**

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

## Remarks

In design mode, this collection is automatically populated with information from the database. In browse mode, this collection contains any schema objects that are used on the specified page, plus any that are explicitly added by the user.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# ScreenUpdating Property

**True** if screen updating is turned on. The default value is **True**. Read/write **Boolean**.

*expression*.**ScreenUpdating**

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

## Remarks

Setting this property to **False** causes the chart workspace or spreadsheet to stop redrawing. To prevent screen flicker or to prevent the user from seeing individual updates, set this property to **False**, perform your update operations, and then reset this property to **True**.

## Example

This example turns off screen updating.

```
Spreadsheet1.ScreenUpdating = False
```



# ScrollColumn Property

-

Returns or sets the number of the leftmost column in the pane or window. If the panes are frozen, this property excludes the frozen areas. Read/write **Long**.

*expression*.**ScrollColumn**

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

## Example

This example moves column C so that it's the leftmost column in the window.

```
Spreadsheet1.ActiveWindow.ScrollColumn = 3
```





# ScrollRow Property

-

Returns or sets the number of the row that appears at the top of the pane or window. If the panes are frozen, this property excludes the frozen areas.  
Read/write **Long**.

*expression*.**ScrollRow**

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

## Example

This example moves row ten to the top of the window.

```
Spreadsheet1.ActiveWindow.ScrollRow = 10
```



# Section Property

-  
Returns a [Section](#) object that represents the section where the specified event occurred.

*expression*.**Section**

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



# Segments Property

-

Returns a **ChSegments** object that represents the collection of segments in the specified format map.

*expression*.**Segments**

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

## Example

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created. The smaller values are displayed in white, then larger values are displayed in a light shade of blue, and finally the larger values in the chart are displayed with in dark blue.

```
Sub Window_Onload()  
  
    Dim serSeries1  
    Dim segSegment1  
    Dim chconstants  
  
    Set chconstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=Da  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind Chartspace1 to the Quanti  
    ' in the Order Details table.  
    ChartSpace1.SetData chconstants.chDimCategories, chconstants.chD  
    ChartSpace1.SetData chconstants.chDimValues, chconstants.chDataE  
  
    ' Create a format map.  
    ChartSpace1.SetData chconstants.chDimFormatValues, chconstants.c  
  
    ' Set a variable to the first series in the first chart in Chart  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a segment to the format map.  
    Set segSegment1 = serSeries1.FormatMap.Segments.Add  
  
    ' Specify that the divisions in formatting be created automatica  
    segSegment1.HasAutoDivisions = True  
  
    ' Measure the segment boundaries based upon a percentage.  
    segSegment1.Begin.ValueType = chconstants.chBoundaryValuePercent  
    segSegment1.End.ValueType = chconstants.chBoundaryValuePercent  
  
    ' Set the beginning value to 0%, and the ending value to 100%.  
    segSegment1.Begin.Value = 0  
    segSegment1.End.Value = 1
```

```
' Format the interior of the matching values.  
segSegment1.Begin.Interior.Color = "White"  
segSegment1.End.Interior.Color = "Blue"
```

End Sub



# SelectedSheets Property

-

Returns a **Sheets** collection that represents all the selected sheets in the specified window. This collection will only contain one sheet, the active sheet.

*expression*.**SelectedSheets**

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



# Selection Property

**ChartSpace** object: Returns an **Object** that represents the selected object. Use the **TypeName** function to determine the type of the selected object.

**PivotTable** object: Returns or sets an **Object** that represents the selected object.

**Spreadsheet** and **Window** objects: Returns a **Range** object that represents the selected cells.

*expression*.**Selection**

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



## Example

The procedure in this example runs whenever the selection in the chart workspace changes. If the user selects an axis, the procedure displays the minimum and maximum values for the axis.

```
Sub ChartSpace1_SelectionChange()  
  
    Dim chConstants  
    Dim minval  
    Dim maxval  
  
    Set chConstants = ChartSpace1.Constants  
  
    If ChartSpace1.SelectionType = chConstants.chSelectionAxis Then  
        minval = ChartSpace1.Selection.Scaling.Minimum  
        maxval = ChartSpace1.Selection.Scaling.Maximum  
  
        MsgBox "minimum = " & minval & " maximum = " & maxval  
    End If  
  
End Sub
```



▼ [Show All](#)

# SelectionType Property

-

**ChartSpace** object: Returns a [ChartSelectionsEnum](#) constant that represents the type of object currently selected in the chart workspace. Read-only.

ChartSelectionsEnum can be one of these ChartSelectionsEnum constants.

**chSelectionAxis**

**chSelectionCategoryLabel**

**chSelectionChart**

**chSelectionChartSpace**

**chSelectionDataLabel**

**chSelectionDataLabels**

**chSelectionDropZone**

**chSelectionErrorbars**

**chSelectionField**

**chSelectionGridlines**

**chSelectionLegend**

**chSelectionLegendEntry**

**chSelectionNone**

**chSelectionPlotArea**

**chSelectionPoint**

**chSelectionSeries**

**chSelectionSurface**

**chSelectionTitle**

**chSelectionTrendline**

**chSelectionUserDefined**

**PivotTable** object: Returns a **String** that represents the type of object currently selected in the PivotTable list. Read-only.

*expression*.**SelectionType**

*expression* Required. An expression that returns a **ChartSpace** or **PivotTable** object.

## Example

The procedure in this example runs whenever the selection in the chart workspace changes. If the user selects an axis, the procedure displays the minimum and maximum values for the axis.

```
Sub ChartSpace1_SelectionChange()  
  
    Dim chConstants  
    Dim minval  
    Dim maxval  
  
    Set chConstants = ChartSpace1.Constants  
  
    If ChartSpace1.SelectionType = chConstants.chSelectionAxis Then  
        minval = ChartSpace1.Selection.Scaling.Minimum  
        maxval = ChartSpace1.Selection.Scaling.Maximum  
  
        MsgBox "minimum = " & minval & " maximum = " & maxval  
    End If  
  
End Sub
```



# Separator Property

-

Returns or sets the string that separates data label components in the specified series or chart. Read/write **String**.

*expression*.**Separator**

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

## Example

This example changes the data-label separator character for the specified series.

```
Sub FormatDataLabels()  
  
    Dim dlDataLabels  
  
    ' Add data labels to the first series.  
    Set dlDataLabels = ChartSpace1.Charts(0).SeriesCollection(0) _  
        .DataLabelsCollection.Add  
  
    ' Display the category name in the data labels.  
    dlDataLabels.HasCategoryName = True  
  
    ' Set the data label separator.  
    dlDataLabels.Separator = ":"  
  
End Sub
```



# SeriesCollection Property

-  
Returns the [ChSeriesCollection](#) collection for the specified chart.

*expression*.**SeriesCollection**

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



## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

## Example

This example sets the line color to red for the specified series.

```
ChartSpace1.Charts(0).SeriesCollection(0).Line.Color = "red"
```



# ServerFilter Property

-  
Returns or sets the server filter for the specified object. Read/write **String**.

*expression*.**ServerFilter**

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

## Remarks

A server filter is a criterion for specifying the type or types of records to be fetched from the server. The data model adds a server filter string to the WHERE clause of the SQL statement that it generates. You can set a server filter only on recordsets that have a primary row source of type **dscTable** or **dscView**.

▼ [Show All](#)

# Sheets Property

## ▸ [Sheets property as it applies to the \*\*Spreadsheet\*\* object.](#)

Returns a [Sheets](#) collection that represents all the sheets in the active workbook.

*expression*.**Sheets**

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

## ▸ [Sheets property as it applies to the \*\*Workbook\*\* object.](#)

Returns a [Sheets](#) collection that represents all the sheets in the specified workbook.

*expression*.**Sheets**

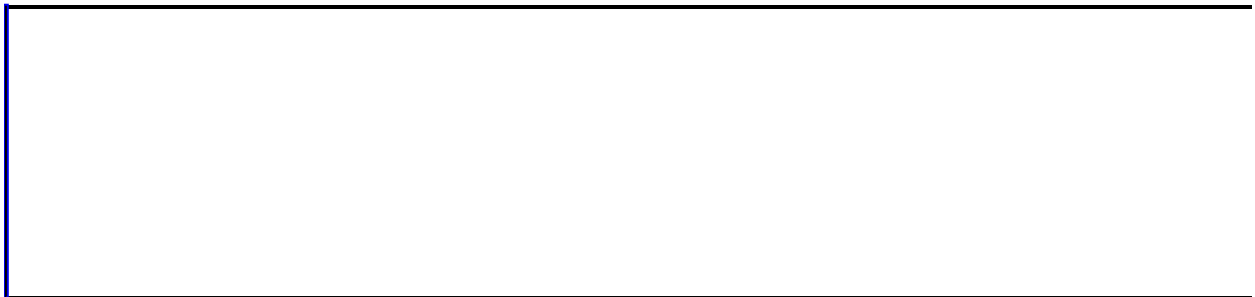
*expression* Required. An expression that returns a **Workbook** object.

# Example

► [As it applies to the \*\*Workbook\*\* object.](#)

This example sorts the worksheets in the active workbook of Spreadsheet1 in ascending order.

```
Sub Sort_Sheets()  
    Dim i  
    Dim j  
    Dim iSheetCount  
  
    iSheetCount = Spreadsheet1.ActiveWorkbook.Sheets.Count  
  
    For i = 1 To iSheetCount  
        For j = 1 To iSheetCount - 1  
            If UCase(Spreadsheet1.ActiveWorkbook.Sheets(j).Name) > _  
                UCase(Spreadsheet1.ActiveWorkbook.Sheets(j + 1).Name) Th  
                Spreadsheet1.ActiveWorkbook.Sheets(j).Move , _  
                    Spreadsheet1.ActiveWorkbook.Sheets(j + 1)  
            End If  
        Next  
    Next  
End Sub
```



# ShowAll Property

-

Clears AutoFilter criteria when set to **True**. Adding criteria sets this property to **False**. If there are no criteria and this property is set to **False**, no data will be shown. Read/write **Boolean**.

*expression*.**ShowAll**

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



## Example

This example clears the AutoFilter criteria for column 1 on the active worksheet and reapplies the AutoFilter to the worksheet.

```
Sub ReapplyAutoFilter()  
    Dim afAutoFilter  
  
    ' Set a variable to the current filter settings.  
    Set afAutoFilter = Spreadsheet1.ActiveSheet.AutoFilter  
  
    ' Show all records.  
    afAutoFilter.Filters(1).Criteria.ShowAll = True  
  
    ' Reapply the filters.  
    afAutoFilter.Apply  
End Sub
```



▼ [Show All](#)

# ShowAs Property

-

Returns or sets a [PivotShowAsEnum](#) constant that represents how a **PivotTotal** object is displayed. Read-write.

PivotShowAsEnum can be one of these PivotShowAsEnum constants.

**plShowAsNormal** *Default*

**plShowAsPercentOfColumnParent** Display the totals as a percentage of the total of each item's parent column.

**plShowAsPercentOfColumnTotal** Display the totals as a percentage of the total of each item's column.

**plShowAsPercentOfGrandTotal** Display the totals as a percentage of the grand total of all the data.

**plShowAsPercentOfRowParent** Display the totals as a percentage of the total of each item's parent row.

**plShowAsPercentOfRowTotal** Display the totals as a percentage of the total of each item's row.

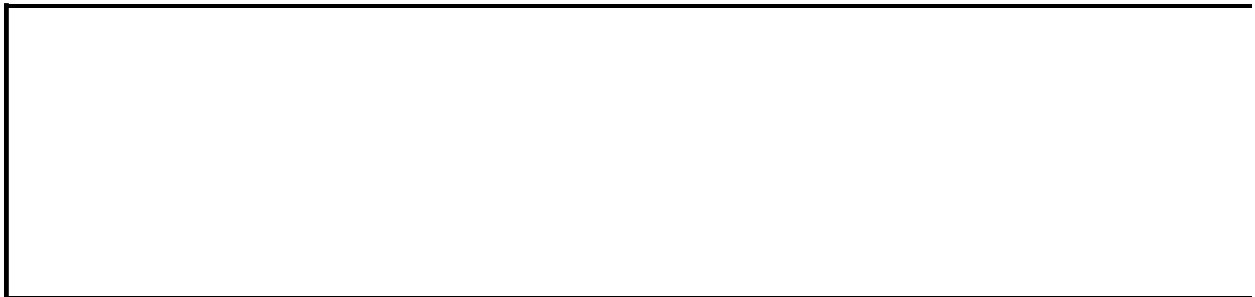
*expression*.**ShowAs**

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

## Example

This example adds a new total to PivotTable1. The new total is formatted to display as a percentage of the parent row field, and will not appear in the PivotTable Field List dialog box.

```
Sub Add_Total()  
  
    Dim vwView  
    Dim ptConstants  
    Dim totNewTotal  
  
    Set vwView = PivotTable1.ActiveView  
    Set ptConstants = PivotTable1.Constants  
  
    ' Add a new total named "Total Budget" to the current view.  
    Set totNewTotal = vwView.AddTotal("Total Budget", vwView.FieldSe  
        ptConstants.plFunctionSum)  
  
    ' Insert the newly created total into the detail area of the Piv  
    vwView.DataAxis.InsertTotal totNewTotal  
  
    ' Show the totals as a percentage of the parent row field.  
    totNewTotal.ShowAs = ptConstants.plShowAsPercentOfRowParent  
  
    ' Do not display the new total in the PivotTable Field List dial  
    totNewTotal.DisplayInFieldList = False  
  
End Sub
```



# ShowAsValue Property

-

Returns a **Variant** that represents the value of the specified **PivotAggregate** object without percentage formatting. Use this property to return the value of a cell's aggregate when the **ShowAs** property has been set to one of the following values: **plShowAsPercentOfColumnParent**, **plShowAsPercentOfColumnTotal**, **plShowAsPercentOfGrandTotal**, **plShowAsPercentOfRowParent**, **plShowAsPercentOfRowTotal**. Read-only.

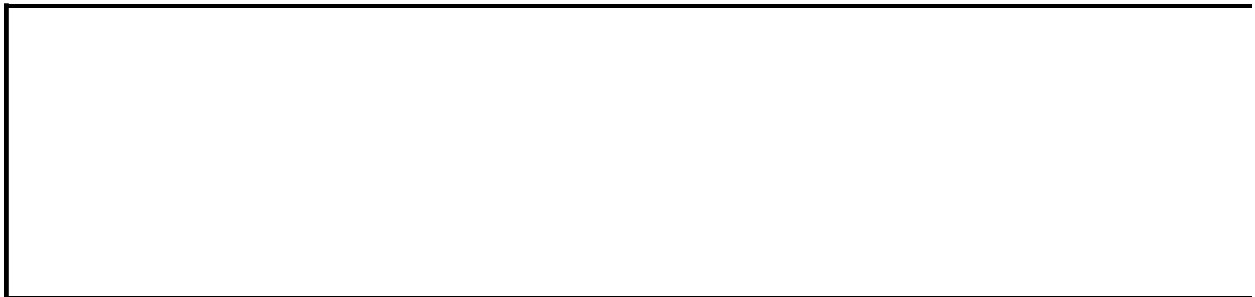
*expression*.**ShowAsValue**

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

## Example

This example displays the aggregate for the third member of the row field in a message box.

```
Sub GetTotal()  
  
    Dim ptData  
    Dim pmRowMem  
    Dim pmColMem  
  
    ' Set a variable to the PivotTable data.  
    Set ptData = PivotTable1.ActiveData  
  
    ' Set a variable to the third item contained in the field  
    ' that has been added to the row axis.  
    Set pmRowMem = ptData.RowAxis.Member.ChildMembers(2)  
  
    ' In this example, there are no fields on the column axis.  
    Set pmColMem = ptData.ColumnAxis.Member  
  
    ' Display the value of the aggregate.  
    MsgBox ptData.Cells(pmRowMem, pmColMem).Aggregates(0).ShowAsValue  
  
End Sub
```



# ShowDelButton Property

**True** if the **Delete** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowDelButton**

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

--

# ShowFilterBySelectionButton Property

**True** if the **Filter by Selection** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowFilterBySelectionButton**

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

---

---



# ShowFirstButton Property

**True** if the **First** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowFirstButton**

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

---

---

# ShowHelpButton Property

**True** if the **Help** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowHelpButton**

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

---

---

# ShowLabel Property

**True** if the record navigation control label is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowLabel**

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

---

---

# ShowLastButton Property

**True** if the **Last** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowLastButton**

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

---

---

# ShowNewButton Property

**True** if the **New** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowNewButton**

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

---

---

# ShowNextButton Property

**True** if the **Next** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowNextButton**

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

---

---

# ShowPrevButton Property

**True** if the **Previous** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowPrevButton**

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

---

---

# ShowSaveButton Property

**True** if the **Save** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowSaveButton**

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

---

---



# ShowSortAscendingButton Property

**True** if the **Sort Ascending** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowSortAscendingButton**

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

---

---

# ShowSortDescendingButton Property

**True** if the **Sort Descending** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowSortDescendingButton**

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

---

---

# ShowToggleFilterButton Property

**True** if the **Toggle Filter** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowToggleFilterButton**

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

---

---

# ShowUndoButton Property

**True** if the **Undo** button (record navigation control) is displayed. The default value is **True**. Read/write **Boolean**.

*expression*.**ShowUndoButton**

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

---

---

# SideWall Property

-

Returns a **ChSurface** object that represents the side wall of a three-dimensional chart. Use the properties and methods of the returned **ChSurface** object to format the side wall of the specified chart.

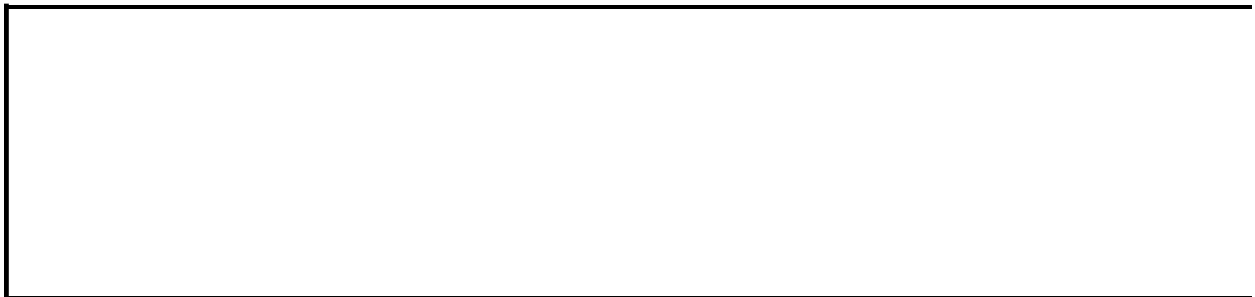
*expression*.**SideWall**

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

## Example

This example converts the first chart in Chartspace1 to a 3-D Column chart and then formats the back wall, side wall, and floor of the chart.

```
Sub FormatWallsFloor()  
  
    Dim cht3DColumn  
    Dim chConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DColumn = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Column chart.  
    cht3DColumn.Type = chConstants.chChartTypeColumnClustered3D  
  
    ' Format the back wall of the chart.  
    cht3DColumn.BackWall.Interior.SetSolid "Yellow"  
    cht3DColumn.BackWall.Thickness = 5  
  
    ' Format the side wall of the chart.  
    cht3DColumn.SideWall.Interior.SetSolid "Yellow"  
    cht3DColumn.SideWall.Thickness = 5  
  
    ' Format the floor of the chart.  
    cht3DColumn.Floor.Interior.SetSolid "Blue"  
    cht3DColumn.Floor.Thickness = 5  
  
End Sub
```



# Size Property

-

Returns or sets the font or marker size (in points). Returns **Null** if it is used on a range in which the characters are not all the same size. Use the **IsNull** function to determine whether the return value is **Null**. Read/write **Variant** for the [Font](#) object; read/write **Long** for all other objects in the Applies To list.

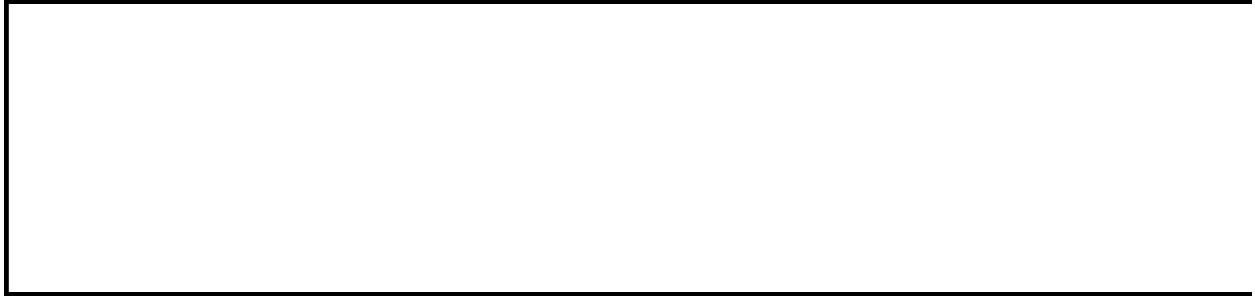
*expression*.**Size**

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

## Example

This example sets the font size to 6 points for the specified axis.

```
Set c = ChartSpace1.Constants  
ChartSpace1.Charts(0).Axes(c.chAxisPositionLeft).Font.Size = 6
```





▼ [Show All](#)

# SizeRepresents Property

-

Returns or sets what the bubble size represents on a bubble chart. Read/write [ChartSizeRepresentsEnum](#).

ChartSizeRepresentsEnum can be one of these ChartSizeRepresentsEnum constants.

**chSizeIsArea**

**chSizeIsWidth**

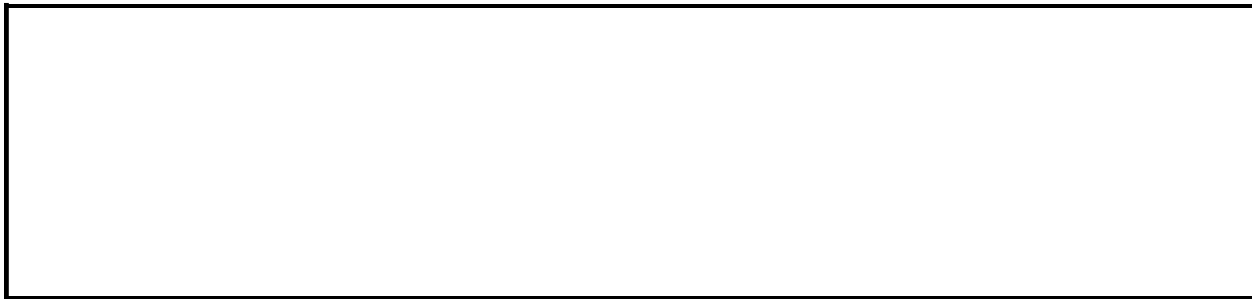
*expression*.**SizeRepresents**

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

## Example

This example sets the **SizeRepresents** property for a bubble chart.

```
Sub SetSizeParameter()  
    Dim chConstants  
    Set chConstants = Chartspace1.Constants  
    ChartSpace1.Charts(0).SizeRepresents = chConstants.chSizeIsArea  
End Sub
```



# SolveOrder Property

-

Returns or sets a **Long** that represents the solve order for the specified total.  
Read/write.

*expression*.**SolveOrder**

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



▼ [Show All](#)

# SortDirection Property

Returns or sets the direction in which the specified field is to be sorted.  
Read/write [PivotFieldSortDirectionEnum](#).

PivotFieldSortDirectionEnum can be one of these PivotFieldSortDirectionEnum constants.

**plSortDirectionAscending**

**plSortDirectionCustom**

**plSortDirectionCustomAscending**

**plSortDirectionCustomDescending**

**plSortDirectionDefault**

**plSortDirectionDescending**

*expression*.**SortDirection**

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

## Remarks

The field is sorted based on the sorting rules of the underlying data provider. For example, a data provider might sort dates alphanumerically (February, January, and so on) or in a chronological monthly sequence (January, February, and so on)

# SortOn Property

-  
Returns or sets the total used to sort the specified field. Read/write [PivotTotal](#).

*expression*.**SortOn**

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

---

---



# SortOnScope Property

-

Returns or sets the array of strings used to narrow the sorting scope for the specified field. Read/write **Variant**.

*expression*.**SortOnScope**

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

--

# Source Property

-

Returns or sets the source for the specified page field or page row source.  
Read/write **String**.

*expression*.**Source**

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

--

# SourceAxis Property

-

Returns a **PivotAxis** object that represents the source axis of the specified result axis.

*expression*.**SourceAxis**

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



# SourceColumnAxis Property

Returns a **PivotGroupAxis** object that represents the source axis of the specified result axis.

*expression*.**SourceColumnAxis**

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



# SourceDataAxis Property

-

Returns a **PivotDataAxis** object that represents the source axis of the specified result axis.

*expression*.**SourceDataAxis**

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



# SourceField Property

-

Returns a **PivotField** object that represents the source field for the specified result field.

*expression*.**SourceField**

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

--

# SourceFilterAxis Property

-

Returns a **PivotFilterAxis** object that represents the source axis of the specified result axis.

*expression*.**SourceFilterAxis**

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



# SourceLabel Property

-

Returns a **PivotLabel** object that represents the source label of the specified result label.

*expression*.**SourceLabel**

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





# SourceMember Property

-

Returns a **PivotMember** object that represents the source member for the specified axis member.

*expression*.**SourceMember**

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



# SourcePageAxis Property

-

Returns a **PivotGroupAxis** object that represents the source axis of the specified result axis.

*expression*.**SourcePageAxis**

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



# SourceRowAxis Property

-

Returns a **PivotGroupAxis** object that represents the source axis for the specified result axis.

*expression*.**SourceRowAxis**

*expression* Required. An expression that returns a **PivotResultRowAxis** property.



# SplitMaximum Property

-

If the specified **ChScaling** object has a split, this property returns or sets the maximum value for the split. This value should be greater than the value of the [SplitMinimum](#) property. Read/write **Double**.

*expression*.**SplitMaximum**

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

## Example

This example splits the value axis of the first chart in ChartSpace1 and sets the split minimum and split maximum values. The value axis is split, and values between 1000 and 5000 will not be displayed.

```
Sub Split_Value_Axis()  
    Dim chConstants  
    Dim scValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the scaling object of the value axis.  
    Set scValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
  
    ' Add a split to the value axis.  
    scValueAxis.HasSplit = True  
  
    ' Specify the minimum value of the split.  
    scValueAxis.SplitMinimum = 1000  
  
    ' Specify the maximum value for the split.  
    scValueAxis.SplitMaximum = 5000  
End Sub
```



# SplitMinimum Property

-

If the specified **ChScaling** object has a split, this property returns or sets the minimum value for the split. This value should be less than the value of the [SplitMaximum](#) property. Read/write **Double**.

*expression*.**SplitMinimum**

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

## Example

This example splits the value axis of the first chart in ChartSpace1 and sets the split minimum and split maximum values. The value axis is split, and values between 1000 and 5000 will not be displayed.

```
Sub Split_Value_Axis()  
    Dim chConstants  
    Dim scValueAxis  
  
    Set chConstants = ChartSpace1.Constants  
  
    ' Set a variable to the scaling object of the value axis.  
    Set scValueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisP  
  
    ' Add a split to the value axis.  
    scValueAxis.HasSplit = True  
  
    ' Specify the minimum value of the split.  
    scValueAxis.SplitMinimum = 1000  
  
    ' Specify the maximum value for the split.  
    scValueAxis.SplitMaximum = 5000  
End Sub
```



# StandardHeight Property

-

Returns the standard (default) height of all the rows in the worksheet in points.  
Read-only **Double**.

*expression*.**StandardHeight**

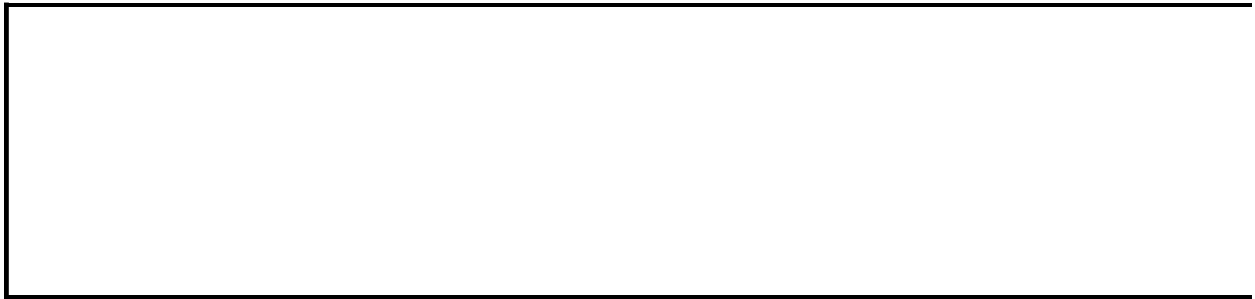
*expression* Required. An expression that returns a **Worksheet** object.



## Example

This example resets the rows and columns in the active sheet of Spreadsheet1 to their default size.

```
Sub Reset_Height_Width()  
    Dim shtActive  
    Set shtActive = Spreadsheet1.ActiveSheet  
  
    shtActive.Rows.RowHeight = shtActive.StandardHeight  
    shtActive.Columns.ColumnWidth = shtActive.StandardWidth  
End Sub
```



# StandardWidth Property

-

Returns the standard (default) width of all the columns in the worksheet. The width of one character in the Normal style is used as the unit of measure. Read/write **Double**.

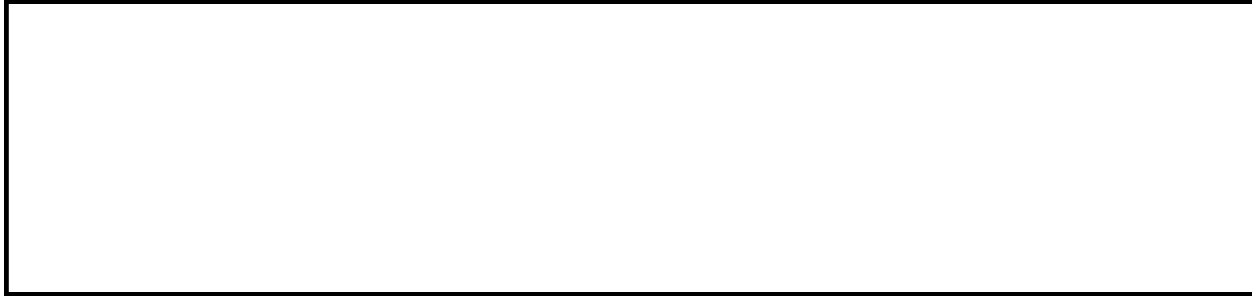
*expression*.**StandardWidth**

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

## Example

This example resets the rows and columns in the active sheet of Spreadsheet1 to their default size.

```
Sub Reset_Height_Width()  
    Dim shtActive  
    Set shtActive = Spreadsheet1.ActiveSheet  
  
    shtActive.Rows.RowHeight = shtActive.StandardHeight  
    shtActive.Columns.ColumnWidth = shtActive.StandardWidth  
End Sub
```



▼ [Show All](#)

# Status Property

-

Returns a [DscStatusEnum](#) constant that represents the status of the current event. This property is supported only in the AfterDelete event. Read-only.

DscStatusEnum can be one of these DscStatusEnum constants.

**dscDeleteCancel** The delete operation succeeded.

**dscDeleteOK** The delete operation was canceled through code.

**dscDeleteUserCancel** The delete operation was canceled by the user.

*expression*.**Status**

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

## Remarks

Using this property with an unsupported event will result in a run-time error.

▼ [Show All](#)

# Style Property

-  
Returns or sets the marker style for the specified series or chart. Read/write [ChartMarkerStyleEnum](#).

ChartMarkerStyleEnum can be one of these ChartMarkerStyleEnum constants.

**chMarkerStyleCircle**

**chMarkerStyleDash**

**chMarkerStyleDiamond**

**chMarkerStyleDot**

**chMarkerStyleNone**

**chMarkerStylePlus**

**chMarkerStyleSquare**

**chMarkerStyleStar**

**chMarkerStyleTriangle**

**chMarkerStyleX**

*expression*.**Style**

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



## Example

This example sets the marker style for the specified series.

```
Sub SetMarkerStyle()  
    Dim chConstants  
    Set chConstants = ChartSpace1.Constants  
    ChartSpace1.Charts(0).SeriesCollection(0).Marker _  
        .Style = chConstants.chMarkerStyleStar  
End Sub
```



# Subaddress Property

-  
Returns or sets a location (within the document) associated with the hyperlink (the string to the right of the # symbol in the hyperlink address). Read/write **String**.

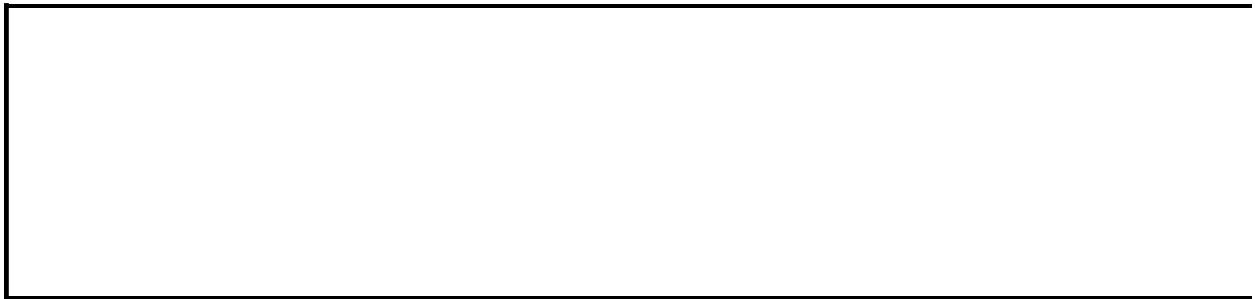
*expression*.**SubAddress**

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

## Example

This example sets the address and subaddress for the specified hyperlink.

```
Sub SetHyperlink()  
    Dim hypHyperlink  
    Set hypHyperlink = Spreadsheet1.Range("A12").Hyperlink  
    hypHyperlink.Address = "http://example.microsoft.com/ExcelDev/e-t  
    hypHyperlink.Subaddress = "top"  
End Sub
```



# SublistRelationships Property

-  
Returns the [SublistRelationships](#) collection for the specified recordset definition.

*expression*.**SublistRelationships**

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

## Remarks

A sublist relationship refers to a row source in another recordset definition that has a many-to-one relationship with the primary page row source in the specified recordset definition.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# SublistSchemaRelationships Property

-  
Returns the [SublistSchemaRelationships](#) collection for the specified schema row source.

*expression*.**SublistSchemaRelationships**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

# SubtotalBackColor Property

-

Returns or sets the back color for the subtotal in the specified field. For subtotals, this property's setting overrides the [TotalBackColor](#) property setting. Read/write **Variant**.

*expression*.**SubtotalBackColor**

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



## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---

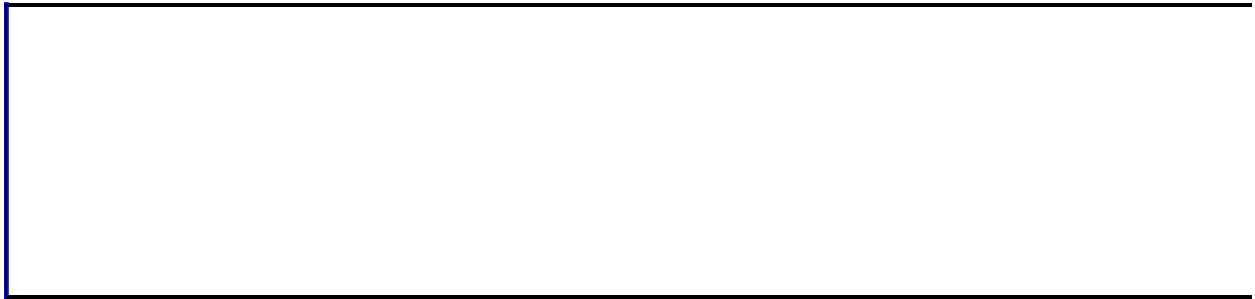
# SubtotalFont Property

-

Returns a [PivotFont](#) object that represents the font for subtotals.

*expression*.**SubtotalFont**

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



# SubtotalForeColor Property

-

Returns or sets the foreground color for subtotals in the specified field. This property's setting overrides the [TotalForeColor](#) property setting. Read/write **Variant**.

*expression*.**SubtotalForeColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---

# SubtotalLabelBackColor Property

-

Returns or sets the back color for the subtotal in the specified field. Read/write **Variant**.

*expression*.**SubtotalLabelBackColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---

# SubtotalLabelFont Property

-

Returns a **PivotFont** object that represents the font for subtotal labels in the specified field.

*expression*.**SubtotalLabelFont**

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



# SubtotalLabelForeColor Property

Returns or sets the foreground color for subtotal labels in the specified field.  
Read/write **Variant**.

*expression*.**SubtotalLabelForeColor**

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



## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---

▼ [Show All](#)

# SubtotalLabelHAlignment Property

Returns or sets a [PivotHAlignmentEnum](#) constant that represents the horizontal alignment of the subtotal labels for the specified field. Read/write.

PivotHAlignmentEnum can be one of these PivotHAlignmentEnum constants.

**plHAlignAutomatic**

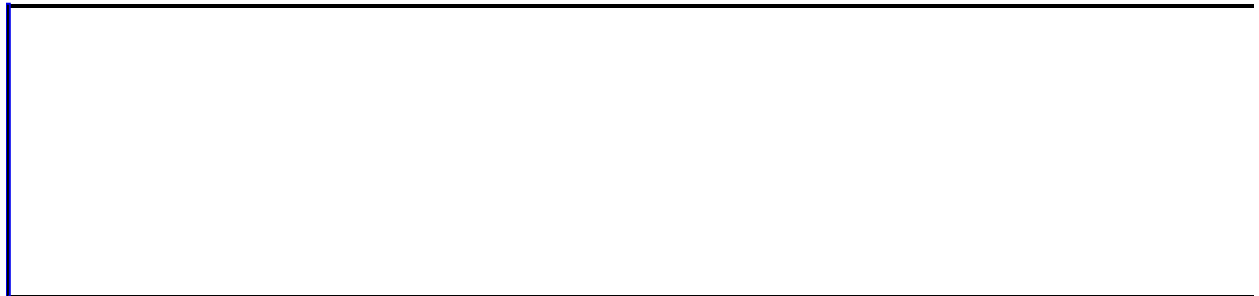
**plHAlignCenter**

**plHAlignLeft**

**plHAlignRight**

*expression*.**SubtotalLabelHAlignment**

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



# Subtotals Property

**True** if the subtotal is displayed for the specified field. Read/write **Boolean**.

*expression*.**Subtotals**(*Subtotals*)

*expression* An expression that returns a [PivotField](#) object. The field must be on a row or column axis.

**Subtotals** Required **Long**. Specifies the subtotal.

---

---

# TabRatio Property

-

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 0.6). Changing the value of this property has an inverse effect on the length of the window's horizontal scroll bar. Increasing this property from the default value decreases the length of the scroll bar, while decreasing this property from the default value increases the length of the scroll bar. Read/write **Double**.

*expression*.**TabRatio**

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

## Example

This example makes the workbook tab half the width of the horizontal scroll bar.

```
Spreadsheet1.ActiveWindow.TabRatio = 0.5
```



▼ [Show All](#)

# Text Property

- ▶ [Text property as it applies to the \*\*PivotAggregate\*\* and \*\*PivotDetailCell\*\* objects.](#)

Returns the specified value as a string based on the current [NumberFormat](#) property setting. Read-only **String**.

*expression*.**Text**

*expression* Required. An expression that returns one of the above objects.

- ▶ [Text property as it applies to the \*\*Range\*\* object.](#)

Returns the formatted value displayed in the specified cell. Returns **Null** if the range includes more than one cell and the cells do not all contain the same value. Use the **IsNull** function to determine whether the return value is **Null**. Read-only **Variant**.

*expression*.**Text**

*expression* Required. An expression that returns a [Range](#) object.



## Example

This example sets a variable for the formatted value of cell A1.

```
txt = Spreadsheet1.Range("a1").Text
```



▼ [Show All](#)

# TextureFormat Property

-

Returns a [ChartTextureFormatEnum](#) constant indicating the format used to display the texture for the specified **ChInterior** object. This property will return a run-time error if the specified interior is not filled with a texture or a picture. Read-only.

ChartTextureFormatEnum can be one of these ChartTextureFormatEnum constants.

**chStack**

**chStackScale**

**chStretch**

**chStretchPlot**

**chTile**

*expression*.**TextureFormat**

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



# TextureName Property

-

Returns a **String** indicating the name of and path to the picture file that was used to fill the specified **ChInterior** object. This property will return a run-time error if the interior of the specified object was set to a preset texture. Read-only.

*expression*.**TextureName**

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

## Remarks

Use the **SetTextured** method to set the texture file for the fill.

▼ [Show All](#)

# TexturePlacement Property

-

Returns a [ChartTexturePlacementEnum](#) constant indicating how the texture has been applied to the specified **ChInterior** object. Read-only.

ChartTexturePlacementEnum can be one of these ChartTexturePlacementEnum constants.

**chAllFaces**

**chEnd**

**chEndSides**

**chFront**

**chFrontEnd**

**chFrontSides**

**chProjectFront**

**chSides**

*expression*.**TexturePlacement**

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

## Remarks

This property only affects 3-D charts.

--



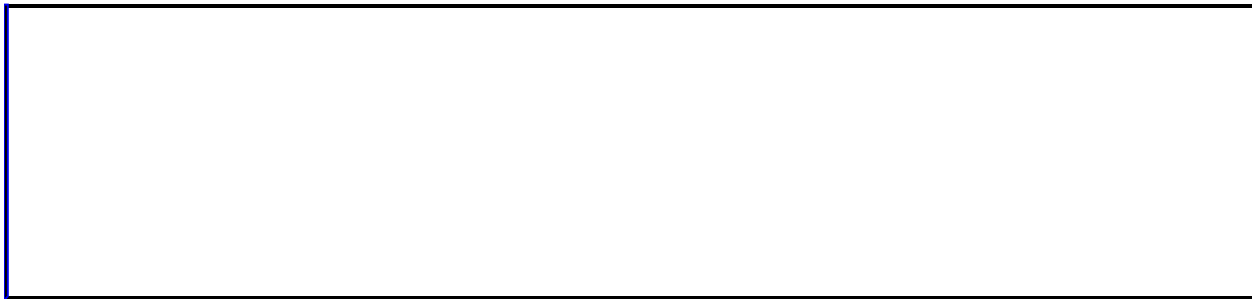
# TextureStackUnit Property

-

Returns a **Double** indicating the texture stack unit for the specified **ChInterior** object. This property will return a run-time error unless the **SetTexture** method was used to fill the interior of the object and the method's *TextureFormat* argument is set to a value of **chStackScale**. Read-only.

*expression*.**TextureStackUnit**

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



# Thickness Property

-

Returns or sets a **Long** specifying the thickness of the specified data series or surface in a three-dimensional chart. Read/write.

*expression*.**Thickness**

*expression* Required. An expression that returns a **ChSeries** or a **ChSurface** object.

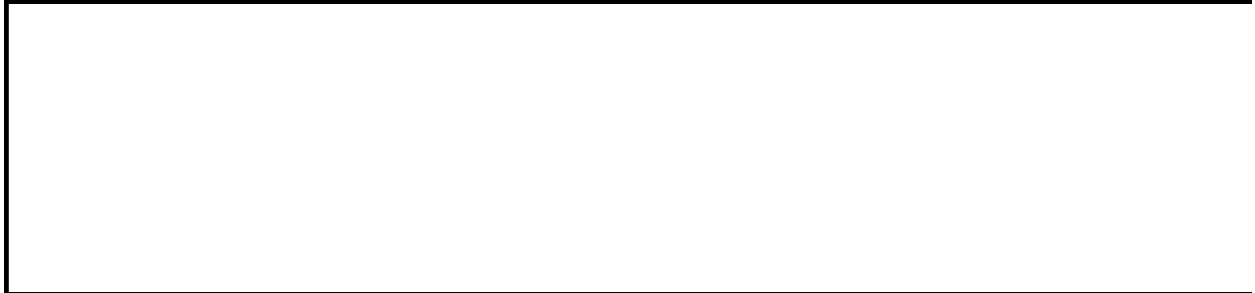
## Remarks

Use this property to set the thickness of a line in a 3-D Line chart or the thickness of the pie in a 3-D Pie chart.

## Example

This example converts the first chart in Chartspace1 to a 3-D Line chart and then sets the thickness for each line in the chart.

```
Sub SetLineThickness()  
  
    Dim cht3DLine  
    Dim serSeries  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set cht3DLine = ChartSpace1.Charts(0)  
  
    ' Change the chart to a 3D Line chart.  
    cht3DLine.Type = chChartTypeLine3D  
  
    ' Set the thickness of each line in the chart.  
    For Each serSeries In cht3DLine.SeriesCollection  
        serSeries.Thickness = 4  
    Next  
End Sub
```



# TickLabelSpacing Property

-

Returns or sets the number of categories between tick-mark labels for the specified axis. Applies only to category axes. Read/write **Long**.

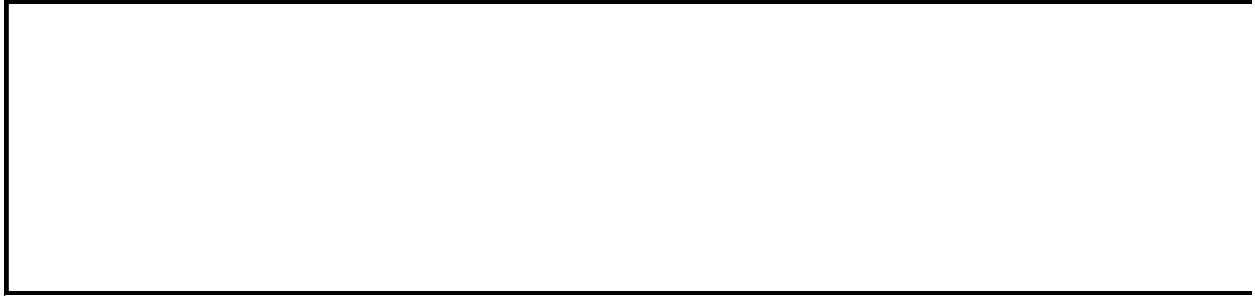
*expression*.**TickLabelSpacing**

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

## Example

This example sets the number of categories between tick-mark labels for the specified axis to two.

```
ChartSpace1.Charts(0).Axes(1).TickLabelSpacing = 2
```



▼ [Show All](#)

# TickLabelUnitType Property

-

Returns or sets a [ChartAxisUnitTypeEnum](#) constant that represents the interval used to display tick mark labels on a time-scaled category axis. Read/write.

ChartAxisUnitTypeEnum can be one of these ChartAxisUnitTypeEnum constants.

**chAxisUnitDay**

**chAxisUnitMonth**

**chAxisUnitQuarter**

**chAxisUnitWeek**

**chAxisUnitYear**

*expression*.**TickLabelUnitType**

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



## Example

This example converts the first chart in Chartspace1 to a line chart, then formats the category axis so that the values are grouped by month. The average value of each month is displayed on the chart.

```
Sub FormatTimeScaling()

    Dim chConstants
    Dim axCategory

    Set chConstants = ChartSpace1.Constants

    ' Change the chart to a Line chart.
    ChartSpace1.Charts(0).Type = chConstants.chChartTypeLine

    ' Set a variable to the category axis.
    Set axCategory = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo

    ' Specify that you will determine the grouping settings of the
    ' axis. Note that this line of code is necessary only if the
    ' GroupingType property for the axis has been previously set to
    ' chAxisGroupingNone.
    axCategory.GroupingType = chConstants.chAxisGroupingManual

    ' Group the category axis by month.
    axCategory.GroupingUnitType = chConstants.chAxisUnitMonth

    ' Create a new grouping for every month.
    axCategory.GroupingUnit = 1

    ' Display the average of the items in each group.
    axCategory.GroupingTotalFunction = chConstants.chFunctionAvg

    ' A tick label is displayed for every month.
    axCategory.TickLabelUnitType = chConstants.chAxisUnitMonth

    ' A tick mark is displayed for every three months.
    axCategory.TickMarkUnitType = chConstants.chAxisUnitQuarter

End Sub
```



# TickMarkSpacing Property

-

Returns or sets the number of categories between tick marks on the specified axis. Applies only to category axes. Use the [MajorUnit](#) and [MinorUnit](#) properties to set tick-mark spacing on value axes. Read/write **Long**.

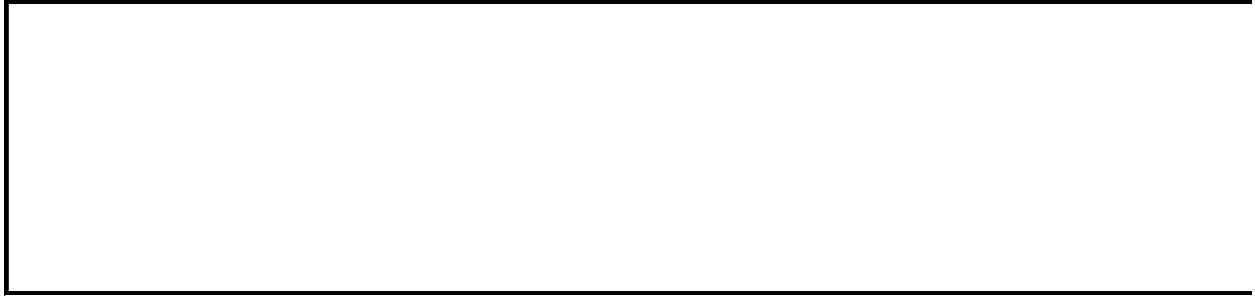
*expression*.**TickMarkSpacing**

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

## Example

This example sets the number of categories between tick marks on the specified axis to two.

```
ChartSpace1.Charts(0).Axes(1).TickMarkSpacing = 2
```



▼ [Show All](#)

# TickMarkUnitType Property

-

Returns or sets a [ChartAxisUnitTypeEnum](#) constant that represents the interval used to display tick marks on a time-scaled category axis. Read/write.

ChartAxisUnitTypeEnum can be one of these ChartAxisUnitTypeEnum constants.

**chAxisUnitDay**

**chAxisUnitMonth**

**chAxisUnitQuarter**

**chAxisUnitWeek**

**chAxisUnitYear**

*expression*.**TickMarkUnitType**

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

## Example

This example converts the first chart in Chartspace1 to a line chart, then formats the category axis so that the values are grouped by month. The average value of each month is displayed on the chart.

```
Sub FormatTimeScaling()

    Dim chConstants
    Dim axCategory

    Set chConstants = ChartSpace1.Constants

    ' Change the chart to a Line chart.
    ChartSpace1.Charts(0).Type = chConstants.chChartTypeLine

    ' Set a variable to the category axis.
    Set axCategory = ChartSpace1.Charts(0).Axes(chConstants.chAxisPo

    ' Specify that you will determine the grouping settings of the
    ' axis. Note that this line of code is necessary only if the
    ' GroupingType property for the axis has been previously set to
    ' chAxisGroupingNone.
    axCategory.GroupingType = chConstants.chAxisGroupingManual

    ' Group the category axis by month.
    axCategory.GroupingUnitType = chConstants.chAxisUnitMonth

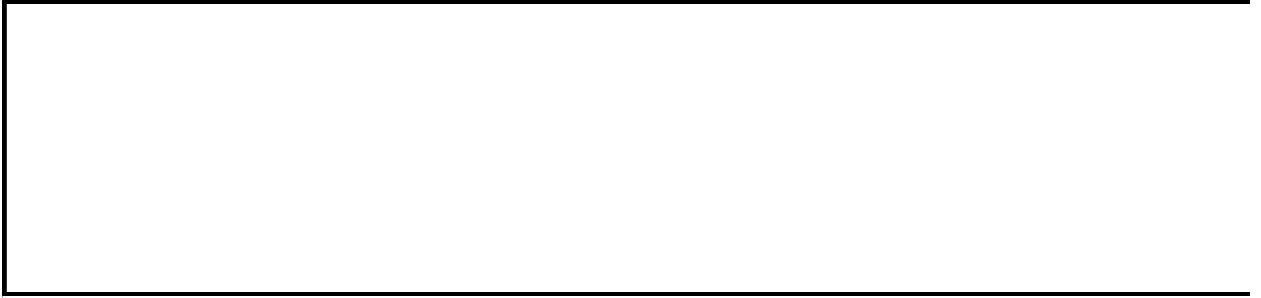
    ' Create a new grouping for every month.
    axCategory.GroupingUnit = 1

    ' Display the average of the items in each group.
    axCategory.GroupingTotalFunction = chConstants.chFunctionAvg

    ' A tick label is displayed for every month.
    axCategory.TickLabelUnitType = chConstants.chAxisUnitMonth

    ' A tick mark is displayed for every three months.
    axCategory.TickMarkUnitType = chConstants.chAxisUnitQuarter

End Sub
```





# TimeValue Property

-

Returns a **Variant** that represents the value of the specified member with the appropriate date of time format. Read-only.

*expression*.**TimeValue**

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

--

▼ [Show All](#)

# TipText Property

► [TipText property as it applies to the \*\*ChSeries\*\* object.](#)

Returns or sets a **String** that represents the chart ScreenTip text for the specified data series. The chart ScreenTip text is formatted with tokens that represent the values associated with the chart ScreenTip. The following table describes the tokens that can be used to set the ScreenTip text. Read/write.

Token	Description
<N>	Where <i>N</i> is a dimension number, returns the name of the dimension. For data series bound to a PivotTable list, returns the name of the pivot
<value>	aggregate. Returns the word "Value" for data series that are not bound to a PivotTable list.
<ls>	Returns the list separator.
 	Inserts a line break.
<<	Returns the "<" symbol.

*expression*.**TipText**

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

► [TipText property as it applies to the \*\*OCCommand\*\* object.](#)

Returns a **String** that represents the ScreenTip text of the specified command. Read-only.

*expression*.**TipText**

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

## Remarks

The **OCCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each Web component.

## Example

► [As it applies to the \*\*ChSeries\*\* object.](#)

This example sets the ScreenTip text of the first series in the first chart in Chartspace1.

```
Sub SetChartTips()  
    ' Set the tiptext for Series1.  
    Chartspace1.Charts(0).SeriesCollection(0) _  
        .TipText = "<1>:<value>:<2>"  
End Sub
```



# Title Property

-  
Returns a [ChTitle](#) object that represents the title of the specified axis or chart.

*expression*.**Title**

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

## Example

This example sets the specified chart to include a title and sets the title text.

```
Sub SetChartTitle()  
    ' Enable the chart title.  
    ChartSpace1.Charts(0).HasTitle = True  
  
    ' Set the chart title.  
    ChartSpace1.Charts(0).Title.Caption = "Satisfaction Data"  
End Sub
```



▼ [Show All](#)



# TitleBar Property

▸ [TitleBar property as it applies to the \*\*PivotView\*\* object.](#)

Returns a [PivotLabel](#) object that represents the title of the specified view.

*expression*.**TitleBar**

*expression* Required. An expression that returns a [PivotView](#) object.

▸ [TitleBar property as it applies to the \*\*Spreadsheet\*\* object.](#)

Returns a [TitleBar](#) object that represents the title of the specified spreadsheet.

*expression*.**TitleBar**

*expression* Required. An expression that returns a [Spreadsheet](#) object.

## Example

This example sets the title caption for the spreadsheet.

```
Spreadsheet1.TitleBar.Caption = "Monthly Sales"
```



# Toolbar Property

-  
Returns an **MSComctlLib.Toolbar** object that represents the toolbar.

*expression*.**Toolbar**

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

## Remarks

Use the returned **MSComctlLib.Toolbar** object to customize the toolbar in the Microsoft Office Web Components. You can add or remove built-in buttons as well as custom buttons.

▼ [Show All](#)

# Top Property

▸ [Top property as it applies to the \*\*ChartSpace\*\*, \*\*ChCategoryLabel\*\*, \*\*ChChartField\*\*, \*\*ChDataLabel\*\*, \*\*ChDataLabels\*\*, \*\*ChErrorBars\*\*, \*\*ChLegendEntry\*\*, \*\*ChPoint\*\*, \*\*ChSeries\*\*, \*\*ChTrendline\*\*, \*\*PivotAggregate\*\*, \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotDetailCell\*\*, \*\*PivotPageMember\*\*, \*\*PivotResultAxis\*\*, \*\*PivotResultColumnAxis\*\*, \*\*PivotResultDataAxis\*\*, \*\*PivotResultFilterAxis\*\*, \*\*PivotResultGroupAxis\*\*, \*\*PivotResultLabel\*\*, \*\*PivotResultPageAxis\*\*, \*\*PivotResultRowAxis\*\*, \*\*PivotRowMember\*\*, and \*\*PivotTable\*\* objects.](#)

Returns (or sets, depending on object used) a **Long** that represents the top edge of the specified object. Read/write.

*expression*.**Top**

*expression* Required. An expression that returns one of the above objects.

▸ [Top property as it applies to the \*\*PivotData\*\* object.](#)

Returns or sets a **PivotRowMember** object that represents the member immediately above the topmost visible summary row.

*expression*.**Top**

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

▸ [Left property as it applies to the \*\*Range\*\* object.](#)

Returns a **Variant** that represents the distance from the top edge of the spreadsheet window to the top edge of the specified range. This value can be negative if the range is above the visible range. Read-only.

*expression*.**Top**

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

► [Left property as it applies to the \*\*Window\*\* object.](#)

Returns a **Double** that represents the top edge of the specified window. Read-only.

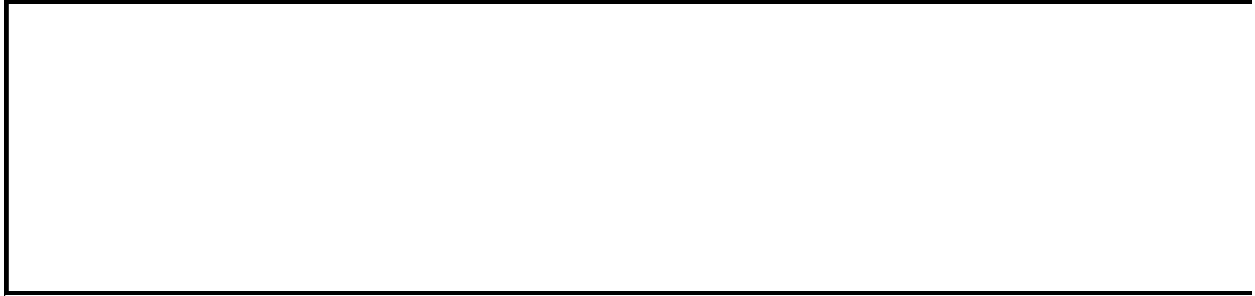
*expression*.**Top**

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

## Example

This example sets a variable for the distance from the top of the spreadsheet window to cell A35.

```
t2 = Spreadsheet1.Range("A35").Top
```





# Top2 Property

-

Returns a **Long** value that represents the top of the data area of the PivotTable list. Read-only.

*expression*.**Top2**

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

## Remarks

This property will return a value of 1 if the toolbar is not displayed.

▼ [Show All](#)

# TopLeft Property

-

► [TopLeft property as it applies to the \*\*PivotRange\*\* object.](#)

Returns a [PivotCell](#) object that represents the upper-left cell in the specified range.

*expression*.**TopLeft**

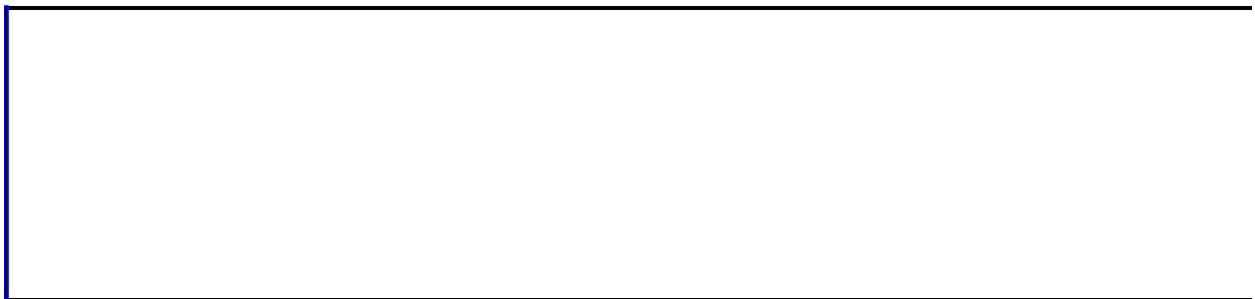
*expression* Required. An expression that returns a [PivotRange](#) object.

► [TopLeft property as it applies to the \*\*PivotDetailRange\*\* object.](#)

Returns a [PivotDetailCell](#) object that represents the upper-left cell in the detail range.

*expression*.**TopLeft**

*expression* Required. An expression that returns a [PivotDetailRange](#) object.



# TopOffset Property

-

Returns or sets a **Long** value that represents the number of pixels to scroll the data area down. Read/write.

*expression*.**TopOffset**

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

## Remarks

Use the **LeftOffset** to scroll the data area to the left.

## Example

This example scrolls the data area of PivotTable 1 down 45 pixels and left 45 pixels.

```
Sub ScrollDataArea()  
    Dim ptData  
    Set ptData = PivotTable1.ActiveData  
    ' Scroll 45 pixels down.  
    ptData.TopOffset = 45  
    ' Scroll the data area to the left.  
    ptData.LeftOffset = 45  
End Sub
```



# Total Property

-  
Returns a [PivotTotal](#) object that represents the total for the specified aggregate.

*expression*.**Total**

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

---

---



# TotalBackColor Property

Returns or sets the background color for all totals. Read/write **Variant**.

*expression*.**TotalBackColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---

# TotalColumnMember Property

-

Returns a **PivotColumnMember** object that represents the member used to display the grand total..

*expression*.**TotalColumnMember**

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



# TotalFont Property

-

Returns a [PivotFont](#) object that represents the font used for aggregates displayed in a cell or in a detail grid footer.

*expression*.**TotalFont**

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

--

# TotalForeColor Property

Returns or sets the foreground color for all totals. Read/write **Variant**.

*expression*.**TotalForeColor**

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

## Remarks

When you set this property, you can use either a **Long** value representing a red-green-blue color value or a **String** value naming a valid HTML color value. For example, to set the object color to red, you could use the hexadecimal value &HFF, the decimal value 255, or the string value "red." In Microsoft Visual Basic, you can use the **RGB** function to create a red-green-blue color value (red is RGB(255, 0, 0)).

This property always returns the color as a **Long** value representing a red-green-blue color value.

---

# TotalMember Property

-

Returns a [PivotAxisMember](#) object that represents the member used to display subtotals.

*expression*.**TotalMember**

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

--

▼ [Show All](#)



# TotalOrientation Property

-

Returns or sets the orientation used to display summary totals when there is more than one total. Read/write [PivotViewTotalOrientationEnum](#).

PivotViewTotalOrientationEnum can be one of these  
PivotViewTotalOrientationEnum constants.

**plTotalOrientationColumn**

**plTotalOrientationRow**

*expression*.**TotalOrientation**

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

---

---

# TotalPageMember Property

-

Returns a **PivotPageMember** object that represents the member used to display the grand total.

*expression*.**TotalPageMember**

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



# TotalRowHeight Property

-

Returns or sets a **Long** value that represents the height of the row that contains the subtotal for the specified member. Read/write.

*expression*.**TotalRowHeight**

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



# TotalRowMember Property

-

Returns a **PivotRowMember** object that represents the member used to display the grand total.

*expression*.**TotalRowMember**

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



# Totals Property

-

**PivotDataAxis** and **PivotResultDataAxis** objects: Returns a **PivotTotals** collection that contains all the **PivotTotal** objects on the summary axis. The totals are displayed in the summary area for each cell.

**PivotView** object: Returns a **PivotTotals** collection that contains all the totals in the current view.

*expression*.**Totals**

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



▼ [Show All](#)

# TotalType Property

-  
Returns or sets the type of total. Read/write [DscTotalTypeEnum](#).

DscTotalTypeEnum can be one of these DscTotalTypeEnum constants.

**dscAny**

**dscAvg**

**dscCount**

**dscMax**

**dscMin**

**dscNone**

**dscStdev**

**dscSum**

*expression*.**TotalType**

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

## Remarks

Values other than **dscNone** are valid only with page fields of type **dscGrouping**.

--



# Trendlines Property

-

Returns the [ChTrendlines](#) collection for the specified series. Note that a series can have only one trendline.

*expression*.**Trendlines**

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

## Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

## Example

This example adds a trendline to the specified series and then hides the trendline's R-squared value.

```
Sub AddTrendline()  
  
    Dim serSeries1  
  
    ' Set a variable to the first series in the first chart of Charts  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a trendline to the series.  
    serSeries1.Trendlines.Add  
  
    ' Hide the R Squared value for the trendline.  
    serSeries1.Trendlines(0).IsDisplayingRSquared = False  
  
End Sub
```



▼ [Show All](#)

# Type Property

▸ [Type property as it applies to the \*\*ChAxis\*\* object.](#)

Returns the object type. Read-only [ChartAxisTypeEnum](#).

ChartAxisTypeEnum can be one of these ChartAxisTypeEnum constants.

**chCategoryAxis**

**chSeriesAxis**

**chTimescaleAxis**

**chValueAxis**

*expression*.**Type**

*expression* Required. An expression that returns one of the above objects.

▸ [Type property as it applies to the \*\*ChChart\*\* and \*\*ChSeries\*\* objects.](#)

Returns or sets the object type. Read/write [ChartChartTypeEnum](#).

ChartChartTypeEnum can be one of these ChartChartTypeEnum constants.

**chChartTypeArea**

**chChartTypeArea3D**

**chChartTypeAreaOverlapped3D**

**chChartTypeAreaStacked**

**chChartTypeAreaStacked100**

**chChartTypeAreaStacked1003D**

**chChartTypeAreaStacked3D**

**chChartTypeBar3D**

**chChartTypeBarClustered**

**chChartTypeBarClustered3D**

**chChartTypeBarStacked**  
**chChartTypeBarStacked100**  
**chChartTypeBarStacked1003D**  
**chChartTypeBarStacked3D**  
**chChartTypeBubble**  
**chChartTypeBubbleLine**  
**chChartTypeColumn3D**  
**chChartTypeColumnClustered**  
**chChartTypeColumnClustered3D**  
**chChartTypeColumnStacked**  
**chChartTypeColumnStacked100**  
**chChartTypeColumnStacked1003D**  
**chChartTypeColumnStacked3D**  
**chChartTypeCombo**  
**chChartTypeCombo3D**  
**chChartTypeDoughnut**  
**chChartTypeDoughnutExploded**  
**chChartTypeLine**  
**chChartTypeLine3D**  
**chChartTypeLineMarkers**  
**chChartTypeLineOverlapped3D**  
**chChartTypeLineStacked**  
**chChartTypeLineStacked100**  
**chChartTypeLineStacked1003D**  
**chChartTypeLineStacked100Markers**  
**chChartTypeLineStacked3D**  
**chChartTypeLineStackedMarkers**  
**chChartTypePie**  
**chChartTypePie3D**  
**chChartTypePieExploded**  
**chChartTypePieExploded3D**  
**chChartTypePieStacked**  
**chChartTypePolarLine**

**chChartTypePolarLineMarkers**  
**chChartTypePolarMarkers**  
**chChartTypePolarSmoothLine**  
**chChartTypePolarSmoothLineMarkers**  
**chChartTypeRadarLine**  
**chChartTypeRadarLineFilled**  
**chChartTypeRadarLineMarkers**  
**chChartTypeRadarSmoothLine**  
**chChartTypeRadarSmoothLineMarkers**  
**chChartTypeScatterLine**  
**chChartTypeScatterLineFilled**  
**chChartTypeScatterLineMarkers**  
**chChartTypeScatterMarkers**  
**chChartTypeScatterSmoothLine**  
**chChartTypeScatterSmoothLineMarkers**  
**chChartTypeSmoothLine**  
**chChartTypeSmoothLineMarkers**  
**chChartTypeSmoothLineStacked**  
**chChartTypeSmoothLineStacked100**  
**chChartTypeSmoothLineStacked100Markers**  
**chChartTypeSmoothLineStackedMarkers**  
**chChartTypeStockHLC**  
**chChartTypeStockOHLC**

*expression*.**Type**

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*ChErrorBars\*\* object.](#)

Returns or sets the object type. Read/write [ChartErrorBarTypeEnum](#).

ChartErrorBarTypeEnum can be one of these ChartErrorBarTypeEnum constants.

**chErrorBarTypeCustom**  
**chErrorBarTypeFixedValue**  
**chErrorBarTypePercent**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*ChScaling\*\* object.](#)

Returns or sets the object type. Read/write [ChartScaleTypeEnum](#).

ChartScaleTypeEnum can be one of these ChartScaleTypeEnum constants.

**chScaleTypeLinear**  
**chScaleTypeLogarithmic**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*ChTrendline\*\* object.](#)

Returns or sets the object type. Read/write [ChartTrendlineTypeEnum](#).

ChartTrendlineTypeEnum can be one of these ChartTrendlineTypeEnum constants.

**chTrendlineTypeExponential**  
**chTrendlineTypeLinear**  
**chTrendlineTypeLogarithmic**  
**chTrendlineTypeMovingAverage**  
**chTrendlineTypePolynomial**  
**chTrendlineTypePower**

*expression.Type*



*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*PageRelationship\*\* object.](#)

Returns the object type. Read-only [DscPageRelTypeEnum](#).

DscPageRelTypeEnum can be one of these DscPageRelTypeEnum constants.

**dscLookup**

**dscSublist**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*SchemaRowsource\*\* object.](#)

Returns or sets the object type. Read/write [DscRowsourceTypeEnum](#).

DscRowsourceTypeEnum can be one of these DscRowsourceTypeEnum constants.

**dscCommandFile**

**dscCommandText**

**dscFunction**

**dscInlineFunction**

**dscProcedure**

**dscTable**

**dscTableFunction**

**dscView**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*PivotFieldSet\*\* object.](#)

Returns the object type. Read-only [PivotFieldSetTypeEnum](#).

PivotFieldSetTypeEnum can be one of these PivotFieldSetTypeEnum constants.

**plFieldSetTypeOther**

**plFieldSetTypeTime**

**plFieldSetTypeUnknown**

**plFieldSetTypeUserDefined**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*PivotField\*\* object.](#)

Returns the object type. Read-only [PivotFieldTypeEnum](#).

PivotFieldTypeEnum can be one of these PivotFieldTypeEnum constants.

**plTypeCalculated**

**plTypeCustomGroup**

**plTypeRegular**

**plTypeTimeDays**

**plTypeTimeHalfYears**

**plTypeTimeHours**

**plTypeTimeMinutes**

**plTypeTimeMonths**

**plTypeTimeQuarters**

**plTypeTimeSeconds**

**plTypeTimeUndefined**

**plTypeTimeWeekdays**

**plTypeTimeWeeks**

**plTypeTimeYears**

**plTypeUnknown**

**plTypeUserDefined**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*PivotTotal\*\* object.](#)

Returns the object type. Read-only [PivotTotalTypeEnum](#).

PivotTotalTypeEnum can be one of these PivotTotalTypeEnum constants.

**plTotalTypeCalculated**

**plTotalTypeIntrinsic**

**plTotalTypeUserDefined**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*Section\*\* object.](#)

Returns the object type. Read-only [SectTypeEnum](#).

SectTypeEnum can be one of these SectTypeEnum constants.

**sectTypeCaption**

**sectTypeFooter**

**sectTypeHeader**

**sectTypeNone**

**sectTypeRecNav**

*expression.Type*

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*Worksheet\*\* object.](#)

Returns the object type. Read-only [XlSheetType](#).

XlSheetType can be one of these XlSheetType constants.

**xlWorksheet**

*expression*.**Type**

*expression* Required. An expression that returns one of the above objects.

► [Type property as it applies to the \*\*Window\*\* object.](#)

Returns the object type. Read-only [XLWindowType](#).

XLWindowType can be one of these XLWindowType constants.

**xlWorkbook**

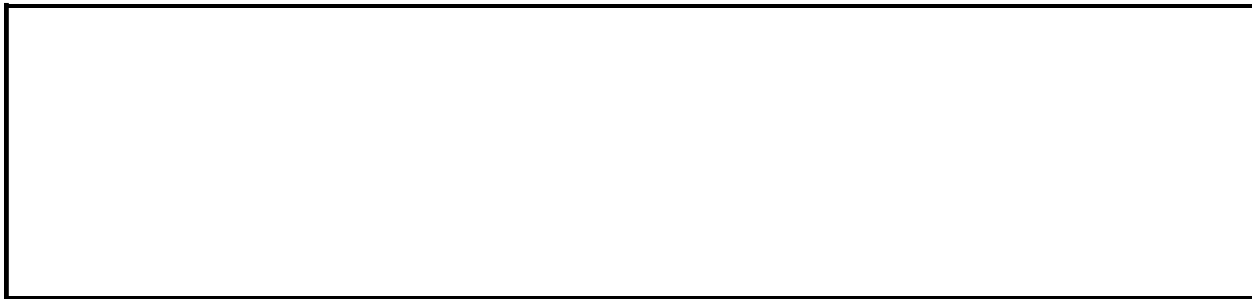
*expression*.**Type**

*expression* Required. An expression that returns one of the above objects.

## Example

This example sets the chart type for the specified chart.

```
Sub SetChartType()  
    Dim chConstants  
    Set chConstants = ChartSpace1.Constants  
    ChartSpace1.Charts(0).Type = chConstants.chChartTypeLineMarkers  
End Sub
```



▼ [Show All](#)

# Underline Property

-

Returns or sets the font underline style for the specified range. Read/write [UnderlineStyleEnum](#) for the [ChFont](#) and [PivotFont](#) objects; read/write **Variant** for the [Font](#) object (returns **Null** if the characters in the font do not all have the same underline style; otherwise, returns one of the **UnderlineStyleEnum** constants). Use the **IsNull** function to determine whether the return value is **Null**.

UnderlineStyleEnum can be one of these UnderlineStyleEnum constants.

**owcUnderlineStyleDouble**

**owcUnderlineStyleDoubleAccounting**

**owcUnderlineStyleNone**

**owcUnderlineStyleSingle**

**owcUnderlineStyleSingleAccounting**

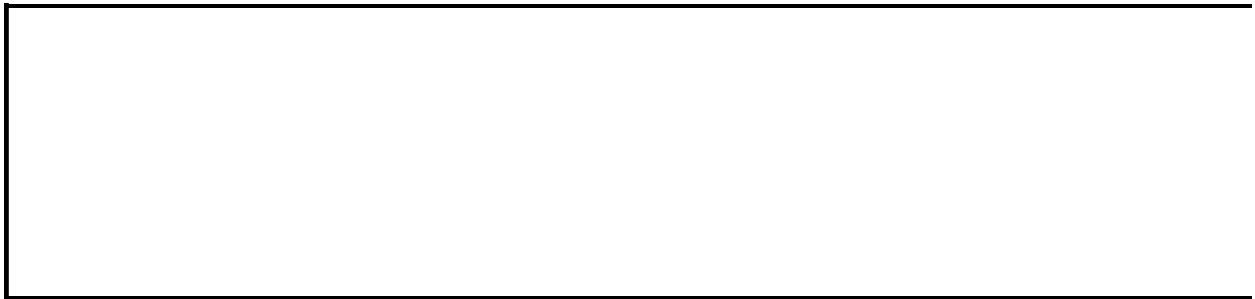
*expression*.**Underline**

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

## Example

This example adds a single underline to all cell values in row 1.

```
Sub FormatFont()  
    Dim ssConstants  
    Set ssConstants = Spreadsheet1.Constants  
    Spreadsheet1.Rows(1).Font.Underline = ssConstants.owcUnderlineSty  
End Sub
```





# UniqueName Property

-

Returns the unique name of the specified object (the unique member reference returned by the provider). Returns **Null** if the member source is a **RecordsetDef** object. Read-only **String**.

*expression*.**UniqueName**

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

--

# UniqueTable Property

-  
Specifies the name of the updatable table when a form is bound to a multitable view or stored procedure. Read/write **String**.

*expression*.**UniqueTable**

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

--

# UsableHeight Property

-

Returns the maximum height of the space in pixels that a window can occupy in the application window area. Returns the same value as the **Height** property. Read-only **Double**.

*expression*.**UsableHeight**

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

## Remarks

You can use the **UsableWidth** property to return the maximum possible width for a window.

# UsableWidth Property

-

Returns the maximum width of the space in pixels that a window can occupy in the application window area. Returns the same value as the **Width** property. Read-only **Double**.

*expression*.**UsableWidth**

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

## Remarks

Use the **UsableHeight** property to return the maximum possible height for a window.

# UsedRange Property

-

Returns a [Range](#) object that represents the used range on the specified worksheet.

*expression*.**UsedRange**

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

## Example

This example adjusts the row height and column width for the used range on the active worksheet to fit the data contained in the range.

```
Sub AutoFitSpreadsheet()  
    Dim rngUsedRange  
  
    ' Set a variable to the used range in the active sheet.  
    Set rngUsedRange = Spreadsheet1.ActiveSheet.UsedRange  
  
    ' Autofit the rows.  
    rngUsedRange.AutoFitRows  
  
    ' Autofit the columns.  
    rngUsedRange.AutoFitColumns  
  
End Sub
```





# UseRemoteProvider Property

**True** if the data source control is using a remote provider. Read/write **Boolean**.

*expression*.**UseRemoteProvider**

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

## Remarks

When this property is set to **True**, the data source control uses the Microsoft Remote Data Services provider for data connections. You can use this property only with pages that are read from a Microsoft Internet Information Server using an HTTP or HTTPS address. The Microsoft Remote Data Services provider fetches data by passing HTTP or HTTPS requests to IIS, which then makes an OLE DB connection to the database.

# UserMode Property

-

Returns a **Boolean** that indicates whether the PivotTable list is in view-only mode. Read-only.

*expression*.**UserMode**

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

--

# UseStandardHeight Property

-

**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 some of the rows are standard height. Returns **False** when none of the rows are the standard height. Read/write **Variant**.

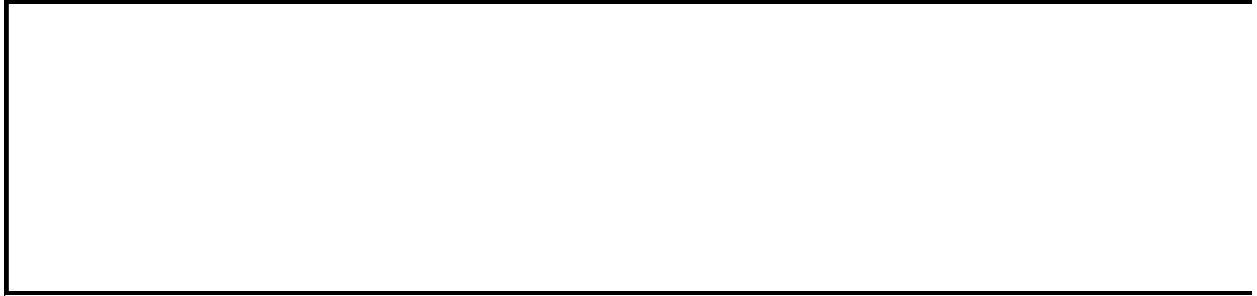
*expression*.**UseStandardHeight**

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

## Example

This example sets the height of row four on Sheet1 in Spreadsheet1 to the standard height.

```
Spreadsheet1.Worksheets("Sheet1").Rows(4).UseStandardHeight = True
```



# UseStandardWidth Property

-

**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 some of the columns are standard width. **False** when none of the columns are the standard width. Read/write **Variant**.

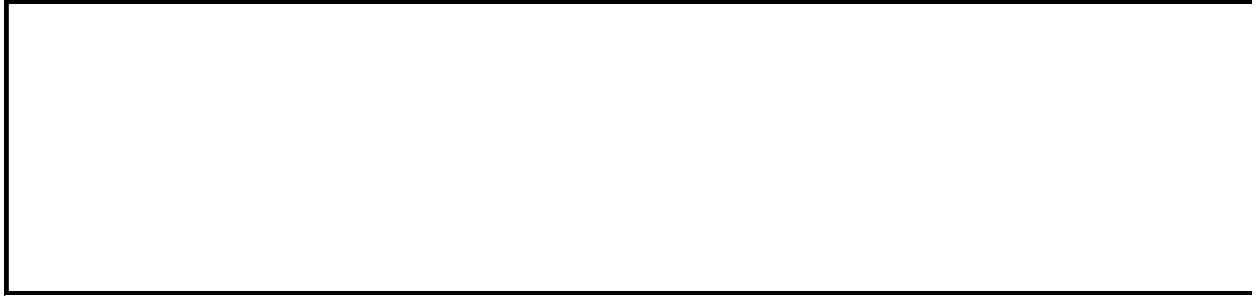
*expression*.**UseStandardWidth**

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

## Example

This example sets the width of column B on Sheet1 in Spreadsheet1 to the standard width.

```
Spreadsheet1.Worksheets("Sheet1").Columns("B").UseStandardWidth = Tr
```



# UseXMLData Property

-  
Returns or sets whether the data access page will bind to XML data. Read/write **Boolean**.

*expression*.**UseXMLData**

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



## Remarks

Use the **XMLLocation** property to set whether the data is located in an XML data island, or a separate XML data file. Use the **XMLDataTarget** property to specify the path or ID to use when binding to the data.

## Example

This example binds the data access page to an XML data file.

```
Sub MSODSC_BeforeInitialBind(DSCEventInfo)

    Dim dscConstants

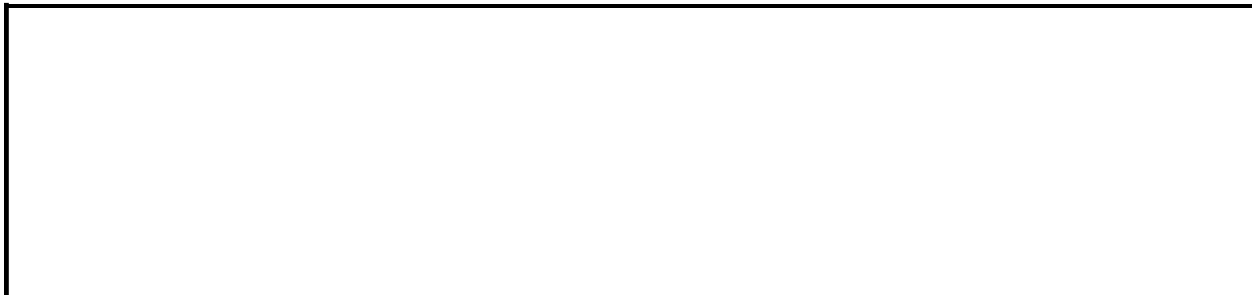
    Set dscConstants = MSODSC.Constants

    ' Set the offline type.
    MSODSC.XMLLocation = dscConstants.dscXMLDataFile

    ' Set the location of the XML data to a data file.
    MSODSC.XMLDataTarget = "Q1 Sales Analysis.xml"

    ' Bind to the XML data.
    MSODSC.UseXMLData = True

End Sub
```



▼ [Show All](#)

# Value Property

## ▸ [Value property as it applies to the \*\*ChSegmentBoundary\*\* object.](#)

Returns or sets a **Double** that represents the value of the specified segment boundary. Read/write.

*expression*.**Value**

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

## ▸ [Value property as it applies to the \*\*Name\*\* and \*\*Spreadsheet\*\* objects.](#)

Returns a **String** that represents the name of the specified object. Read-only.

*expression*.**Value**

*expression* Required. An expression that returns one of the above objects.

## ▸ [Value property as it applies to the \*\*Borders\*\*, \*\*ByRef\*\*, \*\*ParameterValue\*\*, and \*\*PivotDetailCell\*\* objects.](#)

Returns or sets a **Variant** that represents the value of the specified object. Read/write.

*expression*.**Value**

*expression* Required. An expression that returns one of the above objects.

► [Value property as it applies to the \*\*Range\*\* object.](#)

Returns or sets a **Variant** that represents the value of the specified cell.  
Read/write.

*expression*.**Value**(*RangeValueDataType*)

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

**RangeValueDataType** Optional **Variant**. The range value data type.

## Remarks

If the specified range contains more than one cell, this property returns the value of the active cell in the range.

► [Value property as it applies to the \*\*PivotAggregate\*\*, \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotMember\*\*, \*\*PivotPageMember\*\*, \*\*PivotResultMemberProperty\*\*, \*\*PivotRowMember\*\*, and \*\*SchemaProperty\*\* objects.](#)

Returns a **Variant** that represents the value of the specified object. Read-only.

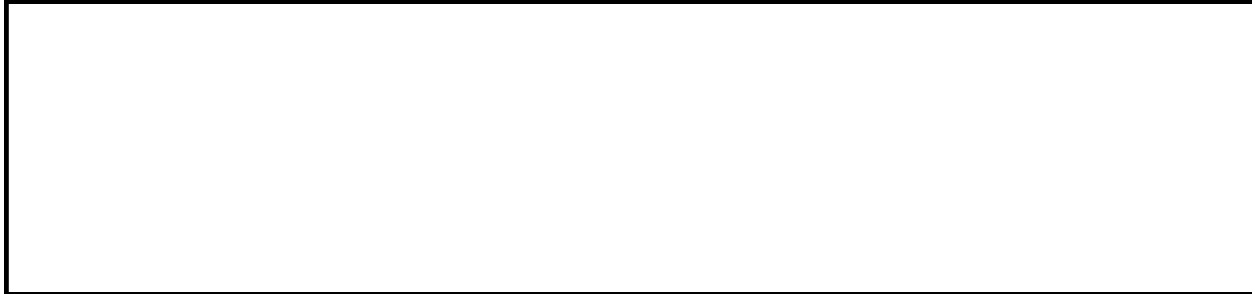
*expression*.**Value**

*expression* Required. An expression that returns one of the above objects.

## Example

This example creates a merged cell from the specified range and then places a value in the merged cell.

```
Sub MergeCells()  
    Spreadsheet1.Range("A1:C3").Merge  
    Spreadsheet1.Range("A1").Value = "Monday"  
End Sub
```



# Value2 Property

-  
Returns or sets a **Variant** representing the cell value. Read/write.

*expression*.**Value2**

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



## 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.

## Example

This example illustrates the differences between the **Value** and the **Value2** properties.

```
Sub Value_vs_Value2()  
    Dim rngCell1  
    Dim rngCell2  
  
    ' Set a variable to the cells used in this example.  
    Set rngCell1 = Spreadsheet1.ActiveSheet.Range("A1")  
    Set rngCell2 = Spreadsheet1.ActiveSheet.Range("A2")  
  
    ' Set the number formats used by the cells in this example.  
    rngCell1.NumberFormat = "Currency"  
    rngCell2.NumberFormat = "Short Date"  
  
    ' Set the value of cell A1 to a currency value.  
    rngCell1.Value = "$123.456789"  
  
    ' Set the value of cell A2 to a date.  
    rngCell2.Value = "9/7/1970"  
  
    ' Use the Value property to return the value of cell A1.  
    MsgBox "Currency returned by the Value Property = " & _  
        rngCell1.Value  
  
    ' Use the Value2 property to return the value of cell A1.  
    MsgBox "Currency returned by the Value2 Property = " & _  
        rngCell1.Value2  
  
    ' Use the Value property to return the value of cell A2.  
    MsgBox "Date returned by the Value Property = " & _  
        rngCell2.Value  
  
    ' Use the Value2 property to return the value of cell A2.  
    MsgBox "Date returned by the Value2 Property = " & _  
        rngCell2.Value2  
End Sub
```



▼ [Show All](#)

# ValueType Property

Returns or sets a [ChartBoundaryValueTypeEnum](#) constant that represents how the **Value** property of the specified segment boundary is interpreted. Read/write.

ChartBoundaryValueTypeEnum can be one of these  
ChartBoundaryValueTypeEnum constants.

**chBoundaryValueAbsolute** The value specified by the **Value** property is a fixed boundary.

**chBoundaryValuePercent** The value specified by the **Value** property is a percentage.

*expression*.**ValueType**

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

## Remarks

When this property is set to **chBoundaryValuePercent**, then the **Value** property of the specified segment boundary must be between 0 and 1.

## Example

This example binds Chartspace1 to the Order Details table in the SQL Server Northwind database. Then, a format map is created that displays the larger values in the chart with a darker shade of blue.

```
Sub Window_Onload()  
  
    Dim serSeries1  
    Dim segSegment1 As ChSegment  
    Dim chconstants  
  
    Set chconstants = ChartSpace1.Constants  
  
    ' The following two lines of code bind Chartspace1 to the Order  
    ' Northwind SQL Server database.  
    ChartSpace1.ConnectionString = "Provider=SQLOLEDB.1;persist Security  
                                   "Catalog=Northwind;Data Source=Da  
    ChartSpace1.DataMember = "Order Details"  
  
    ' The following two lines of code bind Chartspace1 to the Quanti  
    ' in the Order Details table.  
    ChartSpace1.SetData chconstants.chDimCategories, chconstants.chD  
    ChartSpace1.SetData chconstants.chDimValues, chconstants.chDataB  
  
    ' Create a format map.  
    ChartSpace1.SetData chconstants.chDimFormatValues, chconstants.c  
  
    ' Set a variable to the first series in the first chart in Chart  
    Set serSeries1 = ChartSpace1.Charts(0).SeriesCollection(0)  
  
    ' Add a segment to the format map.  
    Set segSegment1 = serSeries1.FormatMap.Segments.Add  
  
    ' Measure the segment boundaries based upon a percentage.  
    segSegment1.Begin.ValueType = chconstants.chBoundaryValuePercent  
    segSegment1.End.ValueType = chconstants.chBoundaryValuePercent  
  
    ' Set the beginning value to 0%, and the ending value to 100%.  
    segSegment1.Begin.Value = 0  
    segSegment1.End.Value = 1  
  
    ' Format the interior of the matching values.  
    segSegment1.Begin.Interior.Color = "White"  
    segSegment1.End.Interior.Color = "Blue"
```

End Sub

--



# Version Property

-

Returns the Microsoft Office Web Components version. Read-only **String**.

*expression*.**Version**

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

▼ [Show All](#)

# VerticalAlignment Property

-

Returns or sets the vertical alignment of the specified object. Can be an [XLVAlign](#) constant. Read/write **Variant**.

XLVAlign can be one of these XLVAlign constants.

**xlVAlignCenter**

**xlVAlignLeft**

**xlVAlignRight**

*expression*.**VerticalAlignment**

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

## Example

This example top-aligns the contents of cells C7:G10 on Sheet1 in Spreadsheet1.

```
Sub SetAlignment()  
    Dim rngAlign  
    Dim ssConstants  
  
    Set ssConstants = Spreadsheet1.Constants  
  
    Set rngAlign = Spreadsheet1.Worksheets("Sheet1").Range("C7:G10")  
  
    ' Center the contents of the range horizontally.  
    rngAlign.HorizontalAlignment = ssConstants.xlHAlignCenter  
  
    ' Vertically align the contents of the range at the top of the ce  
    rngAlign.VerticalAlignment = ssConstants.xlVAlignTop  
End Sub
```



# View Property

-

Returns a **PivotView** object that represents the current view for the specified object.

*expression*.**View**

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

--

# ViewableRange Property

-  
Returns or sets the range of cells that the user can view. This makes it possible to hide worksheet cells (that contain intermediate calculations, for example).

Read/write **String**.

*expression*.**ViewableRange**

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

## Remarks

Do not confuse this property with the [VisibleRange](#) property, which returns a **Range** object that represents all the cells that are currently visible.

## Example

This example sets the viewable range on the spreadsheet.

```
Sub Shrink_Viewable_Range()  
    ' Set the viewable range of the window to cells A1:D10.  
    Spreadsheet1.ActiveWindow.ViewableRange = "A1:D10"  
  
    ' Resize the spreadsheet to eliminate the gray area.  
    Spreadsheet1.Autofit = True  
End Sub
```





# ViewOnlyMode Property

-  
**True** if the Microsoft Office Web Components are in view-only mode. The Web Components will be in view-only mode if the user does not have the appropriate license installed on their computer. Read-only **Boolean**.

*expression*.**ViewOnlyMode**

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

--

# ViewportLeft Property

-

Returns or sets a **Long** value that represents the left side of the viewable data range. Use this property to scroll that data area to the left by a specific number of pixels. Read/write.

*expression*.**ViewportLeft**

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

## Remarks

Use the **ViewportTop** property to return or set the top of the viewable data range.

Setting this property to an invalid value will result in a run-time error. For example, setting this property to 5000 when the data range cannot be scrolled by that many pixels will result in a run-time error.

--

# ViewportTop Property

-

Returns or sets a **Long** value that represents the top of the viewable data range. Use this property to scroll that data area down by a specific number of pixels. Read/write.

*expression*.**ViewportTop**

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

## Remarks

Use the **ViewportLeft** property to return or set the left side of the viewable data range.

Setting this property to an invalid value will result in a run-time error. For example, setting this property to 5000 when the data range cannot be scrolled by that many pixels will result in a run-time error.

▼ [Show All](#)

# Visible Property

-  
**ChDataLabel**, **ChLegendEntry**, **PivotLabel**, and **TitleBar** objects: Returns or sets a **Boolean** that determines whether the specified object is visible. Set this property to **False** to hide the specified object. Read/write.

**Sheets** and **Worksheets** objects: Returns or sets a **Variant** that determines whether the specified object is visible. Set this property to **False** to hide the specified object. Read/write.

**Window** object. Returns a **Boolean** that indicates whether the specified window is visible. Read-only.

**Worksheet** object. Returns or sets a [XlSheetVisibility](#) constant that determines the visibility of the specified worksheet. Read/write.

XlSheetVisibility can be one of these XlSheetVisibility constants.

**xlSheetHidden**

**xlSheetVeryHidden**

**xlSheetVisible**

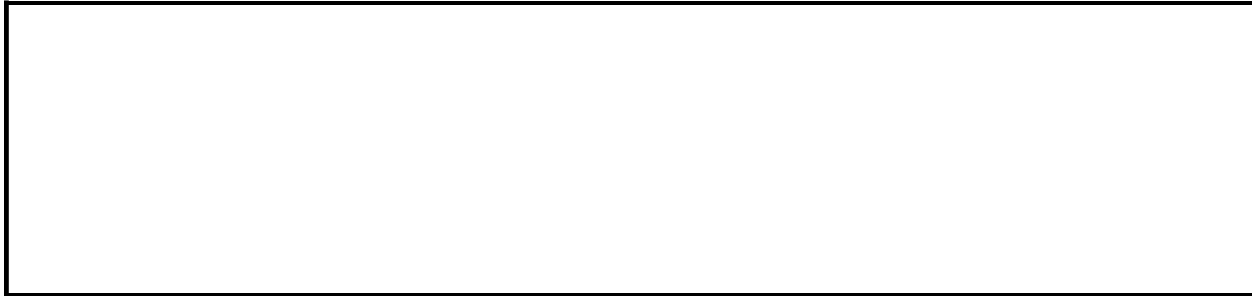
*expression*.**Visible**

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

## Example

This example displays the legend for the specified chart and hides the specified legend entry.

```
Sub ShowLegend()  
    ChartSpace1.Charts(0).HasLegend = True  
    ChartSpace1.Charts(0).Legend.LegendEntries(1).Visible = False  
End Sub
```





# VisibleRange Property

-  
Returns a [Range](#) object that represents all the cells that are currently visible.  
Read-only.

*expression*.**VisibleRange**

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

## Remarks

Do not confuse this property with the [ViewableRange](#) property, which returns a **String** that specifies the range that the user can view (part of that range may not be currently visible).

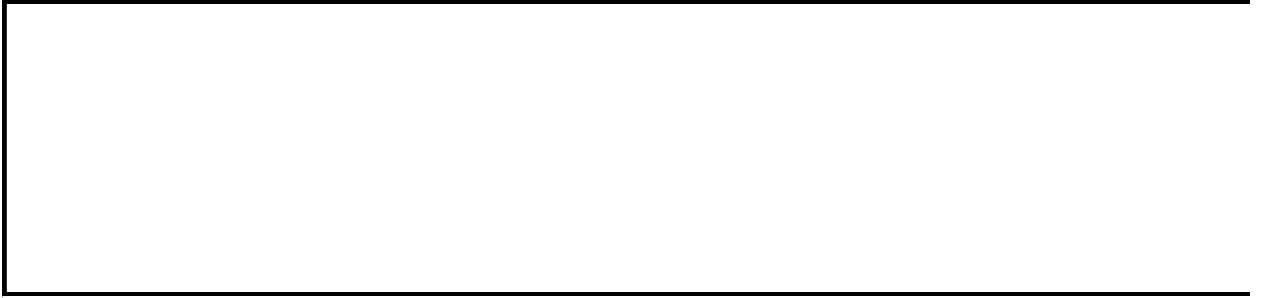
## Example

This example sets the font to bold in every other column in the visible range on the active worksheet.

```
Sub Bold_Odd_Columns()  
    Dim rngColumn  
  
    'Loop through the visible columns.  
    For Each rngColumn In Spreadsheet1.ActiveWindow.VisibleRange.Columns  
  
        'Set the font to bold if the column is odd-numbered.  
        If rngColumn.Column Mod 2 = 0 Then  
            rngColumn.Font.Bold = True  
        End If  
    Next  
End Sub
```

The function in this example returns **True** if the entire current region for cell A1 is visible (if the current region extends outside the visible range, the function returns **False**).

```
Function IsCurrentRegionVisible()  
    Dim rngCurrent  
    Dim rngVisible  
    Dim rngIntersect  
  
    ' Set the variable to the current region of cell A1.  
    Set rngCurrent = Spreadsheet1.ActiveSheet.Cells(1, 1).CurrentRegion  
  
    ' Set a variable to the currently visible range.  
    Set rngVisible = Spreadsheet1.ActiveWindow.VisibleRange  
  
    ' Set a variable to the overlapping portion of the current region  
    ' and the visible range.  
    Set rngIntersect = Spreadsheet1.RectIntersect(rngCurrent, rngVisible)  
  
    ' If the overlapping region is the same as the current region, then  
    ' return true.  
    IsCurrentRegionVisible = (rngIntersect.Address = rngCurrent.Address)  
End Function
```



# WatermarkBorder Property

-

Returns a **ChBorder** object that represents the border of the watermark in the specified drop zone. Use the properties of the returned **ChBorder** object to format the border of the drop zone's watermark.

*expression*.**WatermarkBorder**

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

## Example

This example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```



# WatermarkFont Property

-

Returns a **ChFont** object that represents the font of the watermark in the specified drop zone. Use the properties of the returned **ChFont** object to format the font of the drop zone's watermark.

*expression*.**WatermarkFont**

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

## Example

This example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```





# WatermarkInterior Property

-

Returns a **ChInterior** object that represents the interior of the watermark in the specified drop zone. Use the properties of the returned **ChInterior** object to format the interior of the drop zone's watermark.

*expression*.**WatermarkInterior**

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

## Example

This example formats the button and the watermark of the series drop zone in Chartspace1.

```
Sub SetupDropZone()  
  
    Dim dzSeriesDropZone  
    Dim ChConstants  
  
    Set chConstants = Chartspace1.Constants  
  
    ' Set a variable to the series drop zone in Chartspace1.  
    Set dzSeriesDropZone = ChartSpace1.DropZones(chConstants.chDropZ  
  
    ' The next three lines of code format the button of the drop zon  
    dzSeriesDropZone.ButtonBorder.Weight = chConstants.owcLineWeight  
    dzSeriesDropZone.ButtonInterior.SetSolid "Red"  
    dzSeriesDropZone.ButtonFont.Size = 14  
  
    ' The next three lines of code format the watermark of the drop  
    dzSeriesDropZone.WatermarkBorder.Color = "Red"  
    dzSeriesDropZone.WatermarkFont.Color = "Red"  
    dzSeriesDropZone.WatermarkInterior.SetSolid "Green"  
  
End Sub
```



# Weight Property

-

Returns or sets the weight for the specified border or line. Can be one of the **LineWeightEnum** constants, or can be **Null** if the borders are not all the same weight. Read/write **Variant**.

*expression*.**Weight**

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

## Example

This example sets the axis line and border weight of the chart to thick.

```
Sub FormatChart()  
    Dim chConstants  
    Set chConstants = ChartSpace1.Constants  
    ChartSpace1.Charts(0).Axes(chConstants.chAxisPositionValue).Line  
        .Weight = chConstants.owcLineWeightThick  
    ChartSpace1.Border.Weight = chConstants.owcLineWeightThick  
End Sub
```



▼ [Show All](#)

# Width Property

▸ [Width property as it applies to the \*\*Window\*\* object.](#)

Returns a **Double** that represents the width of the specified window. Read-only.

*expression*.**Width**

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

▸ [Width property as it applies to the \*\*PivotAxisMember\*\*, \*\*PivotColumnMember\*\*, \*\*PivotFieldSet\*\*, \*\*PivotPageMember\*\*, \*\*PivotRowMember\*\*, \*\*PivotTable\*\*, and \*\*PivotTotal\*\* objects.](#)

Returns or sets a **Long** that represents the width of the specified object. Read/write.

*expression*.**Width**

*expression* Required. An expression that returns one of the above objects.

▸ [Width property as it applies to the \*\*PivotAggregate\*\*, \*\*PivotDetailCell\*\*, \*\*PivotResultAxis\*\*, \*\*PivotResultColumnAxis\*\*, \*\*PivotResultDataAxis\*\*, \*\*PivotResultFilterAxis\*\*, \*\*PivotResultGroupAxis\*\*, \*\*PivotResultLabel\*\*, \*\*PivotResultPageAxis\*\*, and \*\*PivotResultRowAxis\*\* objects.](#)

Returns a **Long** that represents the width of the specified object. Read-only.

*expression*.**Width**

*expression* Required. An expression that returns one of the above objects.

► [Width property as it applies to the \*\*Range\*\* object.](#)

Returns a **Variant** that represents the width of the specified object. Read-only.

*expression*.**Width**

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

## Remarks

The **AutoFit** property of the PivotTable list is set to **False** when the value of the **Width** property is changed.



## Example

This example sets the width of the PivotTable list to 150 points.

```
PivotTable1.Object.Width = 150
```



# WidthRatio Property

-  
Returns or sets the width ratio for the specified chart in relation to the other charts in the chart workspace. The default value is 100. Read/write **Long**.

*expression*.**WidthRatio**

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

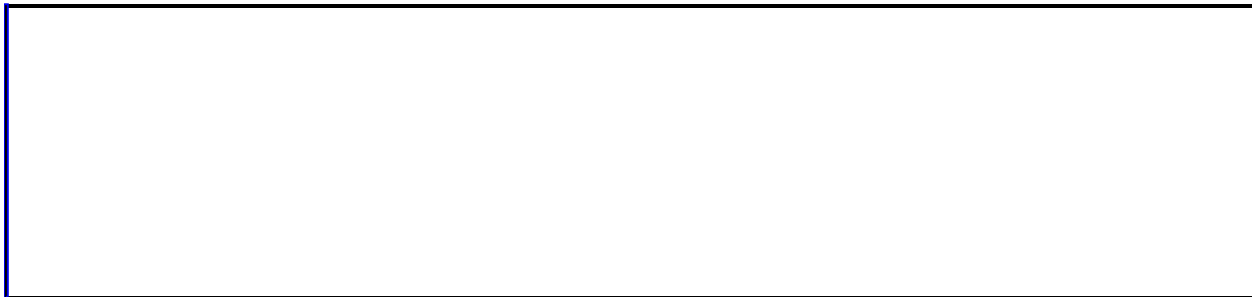
## Remarks

For this property to have any effect, you must have more than one chart in the chart workspace. When more than one chart is displayed, the charts are displayed in a grid (for more information, see the Help topics for the [ChartLayout](#) and [ChartWrapCount](#) properties). Initially, the **HeightRatio** and **WidthRatio** properties are set to 100 for all charts in the grid, and all charts are the same size.

To change the width of charts in the grid, adjust the **WidthRatio** property settings. For example, if each chart is displayed in three columns all charts have an initial **WidthRatio** setting of 100. If you want column 3 to be only half the available column width, set its **WidthRatio** setting to 200; the remaining half of the chart width will be divided between columns 1 and 2. Because the width specified by the **WidthRatio** property is relative, you can set this property for the three columns to 1,1,2; 100,100,200; or 20,20,40. all of which have the same effect.

If the chart workspace contains charts displayed in more than one column, the largest **WidthRatio** setting in each column is used to set the relative width for the entire column.

This property is useful for creating price and volume stock charts in which the volume chart is half the size of the price chart.



# WindowNumber Property

-

Returns the window number. Always returns 1 in this version of the Microsoft Office Spreadsheet Component. Read-only **Long**.

*expression*.**WindowNumber**

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



# Windows Property

-  
Returns a **Windows** collection that represents the windows in the open workbook.

*expression*.**Windows**

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

## Remarks

Although each worksheet in the open workbook has its own window, this property always returns a reference to the active worksheet's window.

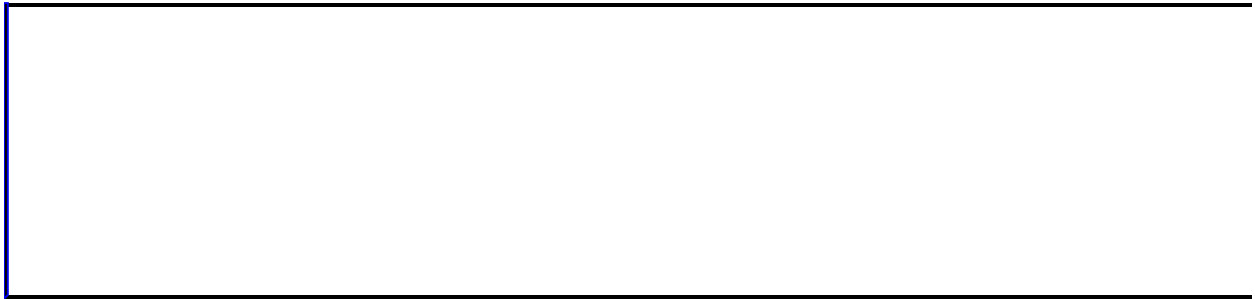
# Workbooks Property

-

Returns a **Workbooks** collection that represents the open workbook.

*expression*.**Workbooks**

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



# Worksheet Property

-

Returns a **Worksheet** object that represents the worksheet containing the specified range. Read-only.

*expression*.**Worksheet**

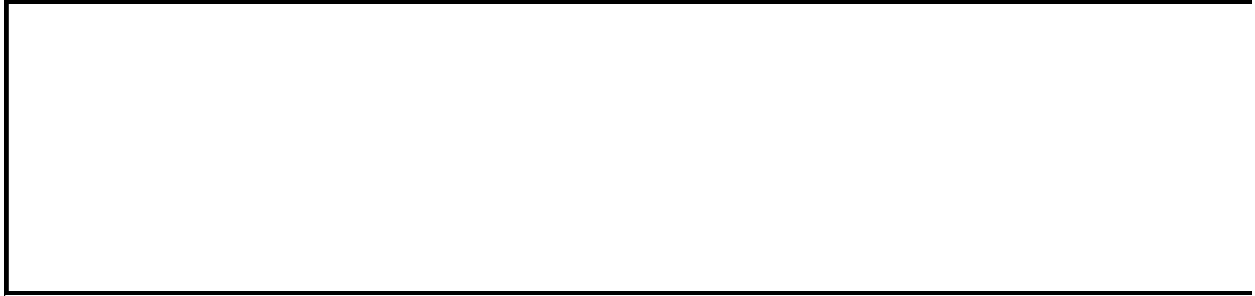
*expression* Required. An expression that returns a **Range** object.



## Example

This example activates the worksheet in Spreadsheet1 that contains the range named "Revenue."

```
Spreadsheet1.Range("Revenue").Worksheet.Activate
```



# Worksheets Property

-  
Returns a **Worksheets** collection that represents the worksheets in the open workbook.

*expression*.**Worksheets**

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

--

# XMLData Property

-  
Returns or sets the XML data. Read/write **String**.

*expression*.**XMLData**

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

--

# XMLDataTarget Property

-

Returns or sets a **String** that represents the location of the XML data to load or save. Read/write.

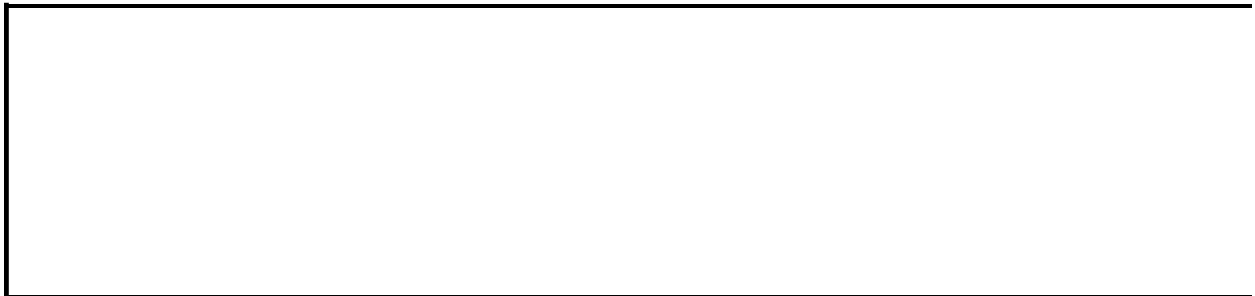
*expression*.**XMLDataTarget**

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

## Example

This example exports the current data in the data source control named MSODSC to an XML data file.

```
Sub ExportData()  
    Dim dscConstants  
  
    Set dscConstants = MSODSC.Constants  
  
    ' Set the location of the XML data to a data file.  
    MSODSC.XMLLocation = dscConstants.dscXMLDataFile  
  
    ' Set the specific target to export to.  
    MSODSC.XMLDataTarget = "Q1 Sales Analysis.xml"  
  
    ' Export the current data.  
    MSODSC.ExportXML  
End Sub
```



▼ [Show All](#)

# XMLLocation Property

-

Returns or sets a [DscXMLLocationEnum](#) constant that specifies whether the XML data is to be loaded or saved from an XML data file or an XML data island inside of the current data access page. Read/write.

DscXMLLocationEnum can be one of these DscXMLLocationEnum constants.

**dscXMLDataFile** The XML data is stored in a separate file.

**dscXMLEmbedded** The XML data is stored in an XML data island located in the current data access page.

*expression*.**XMLLocation**

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

## Example

This example exports the current data in the data source control named MSODSC to an XML data file.

```
Sub ExportData()  
    Dim dscConstants  
  
    Set dscConstants = MSODSC.Constants  
  
    ' Set the location of the XML data to a data file.  
    MSODSC.XMLLocation = dscConstants.dscXMLDataFile  
  
    ' Set the specific target to export to.  
    MSODSC.XMLDataTarget = "Q1 Sales Analysis.xml"  
  
    ' Export the current data.  
    MSODSC.ExportXML  
  
End Sub
```





# XMLURL Property

-  
Returns or sets a **String** representing the URL to an Extensible Markup Language (XML) file. Setting this property discards the currently-open workbook and loads the specified XML file into a new workbook. Read/write.

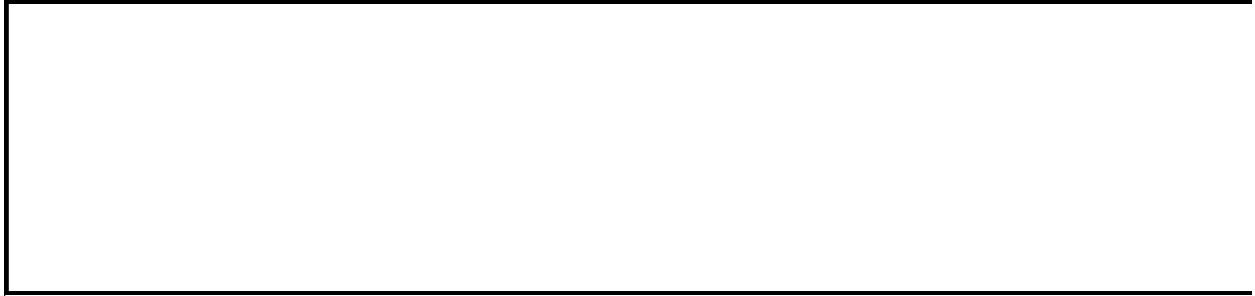
*expression*.**XMLURL**

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

## Example

This example loads the contents of the specified XML file into Sheet1 of Spreadsheet1.

Spreadsheet1.**XMLURL** = "http://example.microsoft.com/Test.xml"



# ZOrder Property

-

Returns or sets a **Long** that specifies the the order in which a series is rendered from front to back. Read/write.

*expression*.**ZOrder**

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

## Remarks

Set this property to 0 to render the series at the front of the chart.

This property affects only clustered or overlapping 3-D Area, Bar, Column, and Line charts.

--

# AfterDelete Event

-

Occurs after a record has been deleted, or the deletion of a record has been canceled. Use this event if you want to perform a set of actions when a record is deleted.

**Private Sub *Object*\_AfterDelete(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Use the **Status** property of the **DSCEventInfo** object to determine whether the record deletion was canceled.

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

## Example

This example displays a message box that indicates the status of the record deletion that fired the event.

```
Sub MSODSC_AfterDelete(DSCEventInfo)

    Dim dscConstants

    Set dscConstants = MSODSC.Constants

    ' Check the status of the record deletion.
    Select Case DSCEventInfo.Status

        ' The record was deleted.
        Case dscConstants.dscDeleteOK

            MsgBox "Record deleted successfully."

        ' The deletion was canceled via code.
        Case dscConstants.dscDeleteCancel

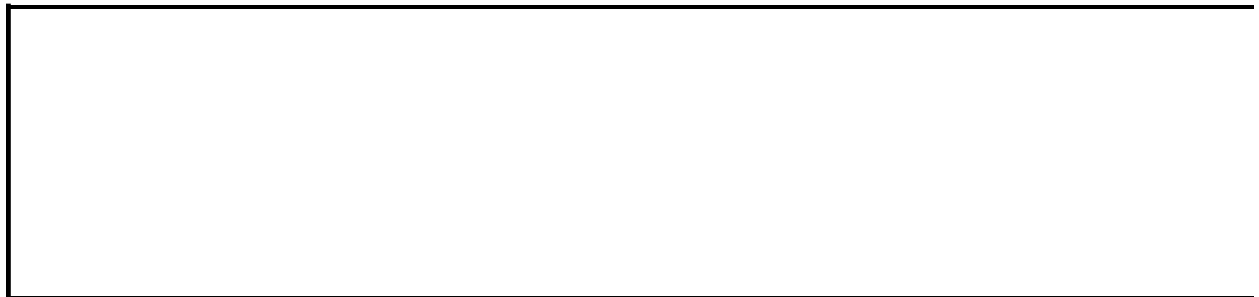
            MsgBox "Record deletion canceled by code."

        ' The deletion was canceled by the user.
        Case dscConstants.dscDeleteUserCancel

            MsgBox "Record deletion canceled by user."

    End Select

End Sub
```



# AfterFinalRender Event

-  
Occurs after all chart elements have been rendered.

**Private Sub ChartSpace\_AfterFinalRender(*drawObject*)**

*drawObject* A **ChChartDraw** object. Use the methods and properties of this object to draw objects on the chart.



## Remarks

You must set the **AllowRenderEvents** property to **True** in order to use this event.

# AfterInsert Event

-

Occurs after a record has been inserted. Use this event if you want to perform a set of actions when a record is inserted.

**Private Sub *Object*\_AfterInsert(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

# AfterLayout Event

-

Occurs after all charts in the specified chart control have been laid out, but before they have been rendered. During this event, you can reposition the **ChTitle**, **ChLegend**, **ChChart**, and **ChAxis** objects of each chart by changing their **Left** and **Top** properties. You can reposition the **ChPlotArea** object by changing its **Left**, **Top**, **Right**, and **Bottom** properties. These properties cannot be changed outside of this event.

**Private Sub ChartSpace\_AfterLayout( ByVal *drawObject* As ChChartDraw)**

***drawObject*** A **ChChartDraw** object. Use the methods and properties of this object to manipulate drawing objects on the chart.

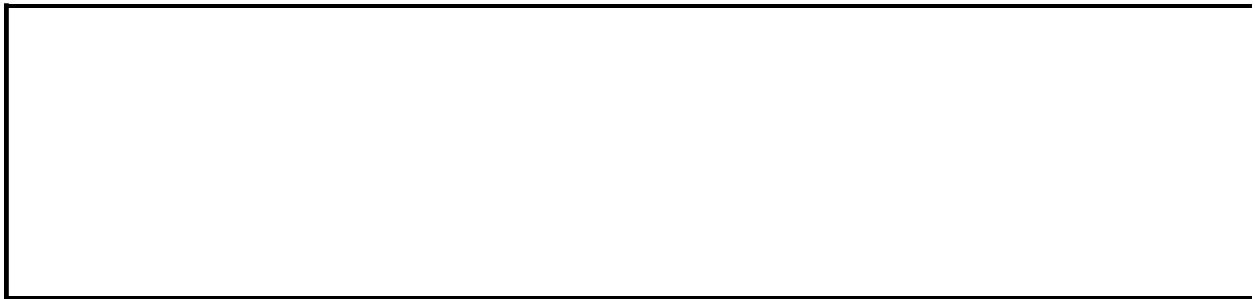
## Remarks

The **AllowLayoutEvents** property must be set to **True** in order to capture this event.

## Example

This example uses the *AfterLayout* event to move the title for the first chart in *Chartspace1* to the left side of the chart. It then moves the legend towards the top of the chart.

```
Private Sub ChartSpace1_AfterLayout()  
    ' Move the title to the left side of the chart.  
    ChartSpace1.Charts(0).Title.Left = 1  
  
    ' Move the legend towards the top of the chart.  
    ChartSpace1.Charts(0).Legend.Top = 20  
End Sub
```



# AfterRender Event

-  
Occurs after the object represented by the *chartObject* argument has been rendered.

**Private Sub ChartSpace\_AfterRender(ByVal *drawObject* As ChChartDraw, ByVal *chartObject* As Object)**

*drawObject* A **ChChartDraw** object. Use the methods and properties of this object to manipulate drawing objects on the chart.

*chartObject* The object that has just been rendered. Use the **TypeName** function to determine what type of object has just been rendered.

## Remarks

You must set the **AllowRenderEvents** and **AllowPointsRenderEvents** properties to **True** in order to use this event with all chart objects.



## Example

This example adds a text string to the upper-left corner of the plot area each time that the chart is redrawn.

```
Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chChart1

    Set chChart1 = ChartSpace1.Charts(0)

    ' After the legend has been rendered, then add the text
    ' to the chart.
    If TypeName(chartObject) = "ChLegend" Then
        drawObject.DrawText "2000 Sales", chChart1.PlotArea.Left + 5
                                chChart1.PlotArea.Top

    End If

End Sub
```

This example illustrates how you can use the BeforeRender and AfterRender events together to create custom gridlines. The BeforeRender event cancels the rendering of the gridlines and the AfterRender event draws custom gridlines.

```
Sub ChartSpace1_BeforeRender(drawObject, chartObject, Cancel)

    ' Check to see if the next object to be rendered
    ' is a gridline.
    If TypeName(chartObject) = "ChGridlines" Then

        ' Cancel the rendering of gridlines.
        Cancel.Value = True

    End If

End Sub

Sub ChartSpace1_AfterRender(drawObject, chartObject)

    Dim chChart1
    Dim plPlotArea
    Dim lLeft
    Dim lRight
    Dim lHeight
```

```

Dim lTop
Dim lIncrement
Dim chConstants
Dim iCtr

Set chConstants = ChartSpace1.Constants

' Set a variable to the first chart in Chartspace1.
Set chChart1 = ChartSpace1.Charts(0)

' Set a variable to the plot area of the chart.
Set plPlotArea = chChart1.PlotArea

' Check to see if the rendered object is a gridline.
If TypeName(chartObject) = "ChGridlines" Then

    ' The next four lines of code use the extents of
    ' the plot area to calculate the dimensions of the line
    ' to be drawn.
    lLeft = plPlotArea.Left
    lTop = plPlotArea.Top
    lRight = plPlotArea.Right
    lHeight = plPlotArea.Bottom - lTop

    ' Determine the increment to use between gridlines.
    ' Change the divisor to adjust the increment.
    lIncrement = lHeight / 10

    ' The next three lines of code set the properties of the
    ' line to be drawn.
    drawObject.Line.DashStyle = chConstants.chLineRoundDot
    drawObject.Line.Color = "Green"
    drawObject.Line.Weight = chConstants.owcLineWeightMedium

    For iCtr = 1 To 9

        ' Draw the line.
        drawObject.DrawLine lLeft, lTop + iCtr * lIncrement, _
            lRight, lTop + iCtr * lIncrement

    Next

End If

End Sub

```



# AfterUpdate Event

-  
Occurs after a record is updated with new data or the record loses focus.

**Private Sub *Object*\_AfterUpdate(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

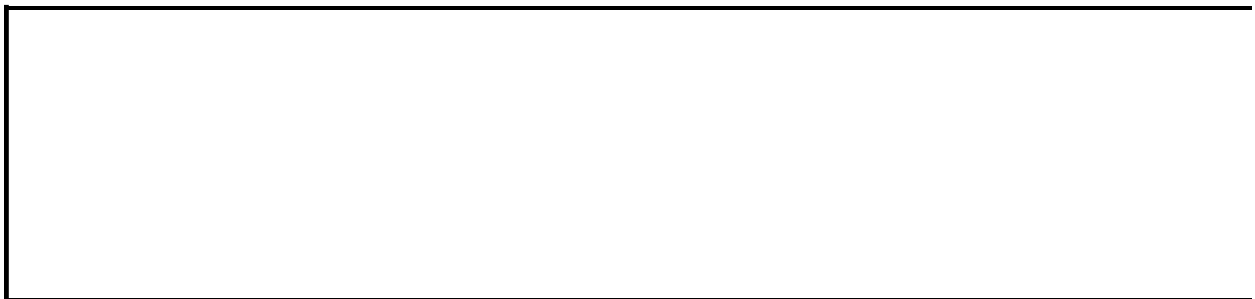
## Remarks

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

## Example

This example displays information about the section that contains the record that was updated.

```
Sub MSODSC_AfterUpdate(DSCInfo)
    MsgBox DSCInfo.Section.HTMLContainer.All(1).InnerText
End Sub
```



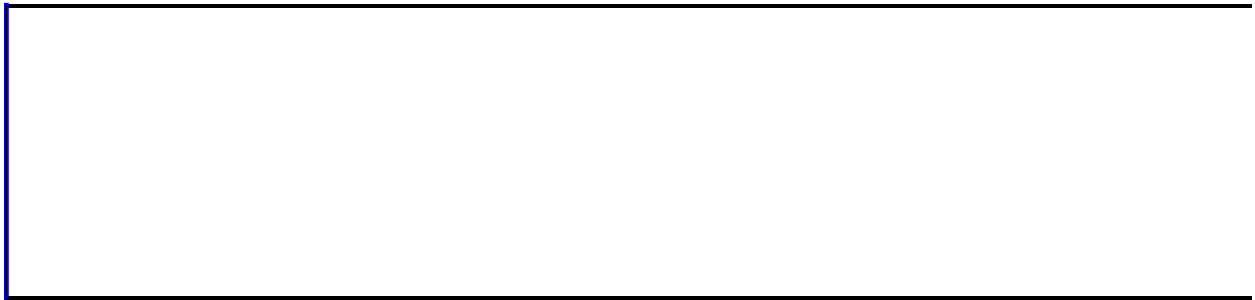
# BeforeCollapse Event

-  
Occurs when the collapse button is clicked on a data access page.

**Private Sub *Object*\_BeforeCollapse(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.



# BeforeContextMenu Event

-

Occurs before a context menu is to be shown. A context menu is shown when the user right-clicks or they press the Application key.

**Private Sub *Object\_BeforeContextMenu*(ByVal *x* As Long, ByVal *y* As Long, ByVal *Menu* As Byref, ByVal *Cancel* As ByRef)**

*x* Represents the x-coordinate where the context menu is to appear.

*y* Represents the y-coordinate where the context menu is to appear.

***Menu*** Set the **Value** property of this object to an array that contains the menu items to display.

***Cancel*** Set the **Value** property of this object to **True** to cancel the keystroke.



## Remarks

Use this event to customize the context menus in the Microsoft Office Web Components.

## Example

This example displays a custom context menu. The menu contains four options, the last option displays a submenu.

```
Sub Spreadsheet1_BeforeContextMenu(x, y, Menu, Cancel)

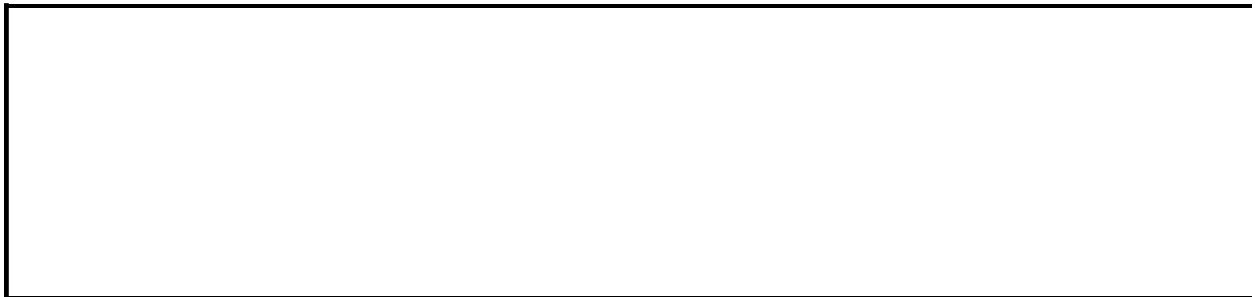
    Dim cmContextMenu(4)
    Dim cmClearSubMenu(2)

    cmClearSubMenu(0) = Array("&All", "ClearAll")
    cmClearSubMenu(1) = Array("&Formats", "ClearFormats")
    cmClearSubMenu(2) = Array("&Values", "ClearValues")

    cmContextMenu(0) = Array("Cu&t", "owc2")
    cmContextMenu(1) = Array("&Copy", "owc3")
    cmContextMenu(2) = Array("&Paste", "owc4")
    cmContextMenu(3) = Empty
    cmContextMenu(4) = Array("Clea&r", cmClearSubMenu)

    Menu.Value = cmContextMenu

End Sub
```



# BeforeDelete Event

-

Occurs before a record is deleted. Use this event if you want to apply a set of conditions before a record is deleted.

**Private Sub *Object*\_BeforeDelete(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Set the **ReturnValue** property of the **DSCEventInfo** object to **False** to cancel the deletion of a record. When you cancel the deletion of a record, the **AfterDelete** event still fires.

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

Use the **DisplayAlert** property to determine whether or not the user is prompted when this event is called.

## Example

This example cancels the deletion of a record if the "Discontinued" field is set the No.

```
Sub MSODSC_BeforeDelete(DSCEventInfo)

    Dim txtDiscontinued

    ' Set a variable to the text box that contains the value
    ' of the Discontinued field for the record that is to be deleted.
    Set txtDiscontinued = DSCEventInfo.Section.HTMLContainer _
        .Children("Discontinued")

    ' Check the value of the control.
    If txtDiscontinued.Value = "No" Then

        ' Display a message to the user.
        MsgBox "Do not delete products that have not " & _
            "been discontinued."

        Cancel the deletion of the record.
        DSCEventInfo.ReturnValue = False
    End If

End Sub
```



# BeforeExpand Event

-  
Occurs whenever the expand button is clicked on a data access page.

**Private Sub *Object*\_BeforeExpand(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

---

---

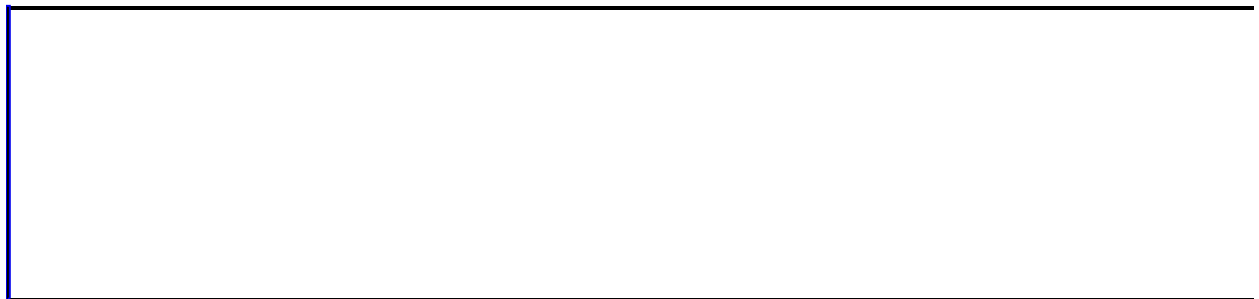
# BeforeFirstPage Event

-  
Occurs before the first set of records is displayed on a banded data access page.

**Private Sub***Object\_BeforeFirstPage*(*DSCEventInfo* As DSCEVENTINFO)

**Object** The name of the **DataSourceControl** object that this event applies to.

**DSCEventInfo** The **DSCEventInfo** object that contains information about the event.



# BeforeInitialBind Event

-

Occurs before the controls on the specified data access page are bound to the recordset for the first time. Use this event to set the properties for the data access page before the controls are populated with data.

**Private Sub *Object*\_BeforeInitialBind(ByVal *DSCEventInfo* As DSCEventInf**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.



## Remarks

This event does not support any of the **DSCEventInfo** properties.

# BeforeInsert Event

-

Occurs when the first character is entered into a new record, but before the record is added to the recordset.

**Private Sub *Object*\_BeforeInsert(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Set the **ReturnValue** property of the **DSCEventInfo** object to **False** to cancel the insertion of a new record.

You can use the **DataPage** property of the **DSCEventInfo** object to get more information about the data access page.

## Example

This example uses the **BeforeInsert** event to prevent the user from adding another record to the recordset once it reaches 75 records.

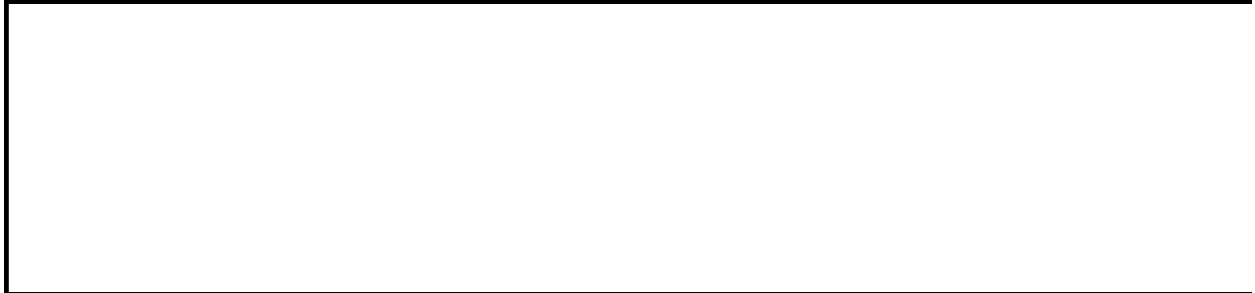
```
Sub MSODSC_BeforeInsert(DSCEventInfo)
    Dim rstCurrentData

    ' Set a variable to the recordset.
    Set rstCurrentData = DSCEventInfo.DataPage.Recordset

    ' Check to see if the recordset has reached its limit.
    If rstCurrentData.RecordCount >= 75 then

        ' Display a message to the user.
        MsgBox "Cannot add any more records."

        ' Cancel the insertion of the record.
        DSCEventInfo.ReturnValue = False
    End If
End Sub
```



# BeforeKeyDown Event

-

Occurs when a user presses a key on the keyboard, but before the control has processed the keystroke. If the user holds the key down, this event repeats itself at the key-repeat interval that has been set on the user's computer.

**Private Sub *Object*\_BeforeKeyDown**(ByVal *KeyCode* As Long, ByVal *Shift* As Long, ByVal *Cancel* As ByRef)

**Object** A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

**KeyCode** An integer that represents the key code of the key that was pressed or released.

**Shift** The state of the SHIFT, CTRL, and ALT keys. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, and 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.

**Cancel** Set the **Value** property of this object to **True** to cancel the keystroke.

## Remarks

Canceling this event also cancels the BeforeKeyPress and KeyPress events, but does not prevent the the KeyDown or KeyUp events from firing.

The sequence of keyboard-related events is:

1. BeforeKeyDown
2. KeyDown
3. BeforeKeyPress
4. KeyPress
5. BeforeKeyUp
6. KeyUp

--

# BeforeKeyPress Event

-

Occurs when a user presses and releases a key on the keyboard, but before the control has processed the keystroke. If the user holds the key down, this event repeats itself at the key-repeat interval that has been set on the user's computer.

**Private Sub *Object*\_BeforeKeyPress**(ByVal *KeyAscii* As Long, ByVal *Cancel* As ByRef)

***Object*** A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

***KeyAscii*** An integer that represents the key code of the key that was pressed or released.

***Cancel*** Set the **Value** property of this object to **True** to cancel the keystroke.

## Remarks

Canceling this event does not prevent the KeyPress or KeyUp events from firing.

The sequence of keyboard-related events is:

1. BeforeKeyDown
2. KeyDown
3. BeforeKeyPress
4. KeyPress
5. BeforeKeyUp
6. KeyUp

--



# BeforeKeyUp Event

-

Occurs when a user releases a key on the keyboard, but before the control has processed the keystroke.

**Private Sub *Object*\_BeforeKeyUp**(ByVal *KeyCode* As Long, ByVal *Shift* As Long, ByVal *Cancel* As ByRef)

***Object*** The name of the **ChartSpace**, **PivotTable**, or **Spreadsheet** object that this event applies to..

***KeyCode*** An integer that represents the key code of the key that was pressed or released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, and 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, or ALT keys were pressed.

***Cancel*** Set the **Value** property of this object to **True** to cancel the keystroke.

## Remarks

Canceling this event does not prevent the KeyUp event from firing.

The sequence of keyboard-related events is:

1. BeforeKeyDown
2. KeyDown
3. BeforeKeyPress
4. KeyPress
5. BeforeKeyUp
6. KeyUp

--

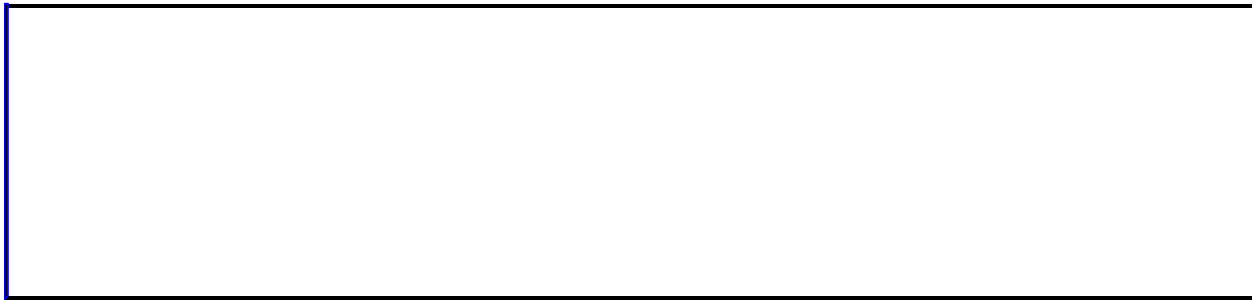
# BeforeLastPage Event

-  
Occurs before the last set of records is displayed on a banded data access page.

**Private Sub *Object*\_BeforeLastPage(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.



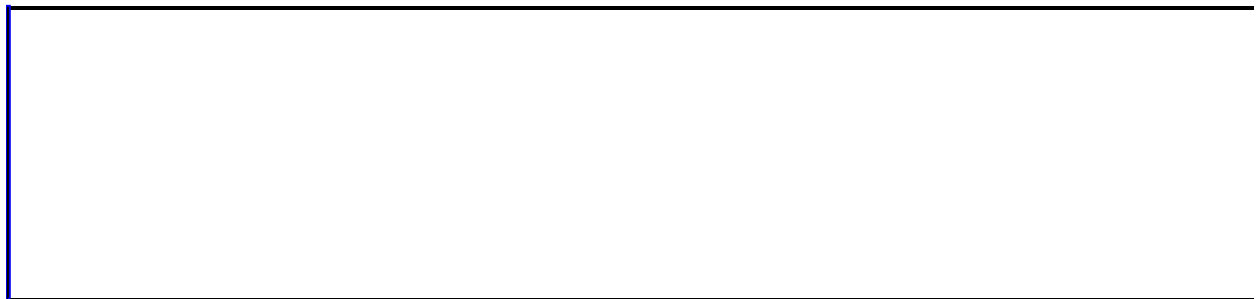
# BeforeNextPage Event

-  
Occurs before the next set of records is displayed on a banded data access page.

**Private Sub *Object*\_BeforeNextPage(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.



# BeforeOverwrite Event

-  
Occurs when an existing file is about to be overwritten.

**Private Sub *Object*\_BeforeOverwrite(ByVal *DSCEventInfo* As DSCEventInfo**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

This event may occur when you use the **ExportXML** method to export the current recordset. Set the **ReturnValue** property of the **DSCEventInfo** object to **False** to prevent the existing file from being overwritten. Set the **DisplayAlert** property of the **DSCEventInfo** object to **dscDataAlertContinue** to overwrite the file without prompting the user.

Note: Files that were not created by the Data Source Control will not be overwritten.

## Example

This example allows a file created by the **ExportXML** method to be overwritten without prompting the user.

```
Sub MSODSC_BeforeOverwrite(DSCEventInfo)

    Dim dscConstants
    Set dscConstants = MSODSC.Constants

    'Don't alert the user when overwriting an existing file.
    DSCEventInfo.DisplayAlert = dscConstants.dscDataAlertContinue

End Sub
```



# BeforePreviousPage Event

-

Occurs before the previous set of records is displayed on a banded data access page.

**Private Sub *Object*\_BeforePreviousPage(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

---

---



# BeforeQuery Event

-  
Occurs when the PivotTable list queries its data source.

**Private Sub *Object*\_BeforeQuery()**

***Object*** The name of the **PivotTable** object that you are trapping this event for.

## Remarks

This event occurs quite frequently. Some examples of actions that trigger this event include adding fields to the PivotTable list, moving fields, sorting, or filtering data.

# BeforeRender Event

-  
Occurs before the object passed in the *chartObject* argument has been rendered.

**Private Sub *Object\_BeforeRender*(ByVal *drawObject* As ChChartDraw, ByVal *chartObject* As Object, *Cancel* As ByRef)**

***Object*** The name of the **ChartSpace** object that you are trapping this event for.

***drawObject*** A reference to the **ChChartDraw** object. Use the **DrawType** property of the returned object to determine what type of rendering is about to occur.

***chartObject*** The object that is to be rendered. Use the **TypeName** function to determine the type of the object.

***Cancel*** Set the **Value** property of this object to **True** to cancel the rendering of the object represented by *chartObject*.

## Remarks

You must set the **AllowRenderEvents** and **AllowPointsRenderEvents** properties to **True** in order to use this event with all chart objects.

## Example

This example illustrates how you can use the BeforeRender and AfterRender events together to create custom gridlines. The BeforeRender event cancels the rendering of the gridlines and the AfterRender event draws custom gridlines.

```
Sub ChartSpace1_BeforeRender(drawObject, chartObject, Cancel)

    ' Check to see if the next object to be rendered
    ' is a gridline.
    If TypeName(chartObject) = "ChGridlines" Then

        ' Cancel the rendering of gridlines.
        Cancel.Value = True
    End If
End Sub

Sub ChartSpace1_AfterRender(drawObject, chartObject)
    Dim chChart1
    Dim plPlotArea
    Dim lLeft
    Dim lRight
    Dim lHeight
    Dim lTop
    Dim lIncrement
    Dim chConstants
    Dim iCtr

    Set chConstants = ChartSpace1.Constants

    ' Set a variable to the first chart in Chartspace1.
    Set chChart1 = ChartSpace1.Charts(0)

    ' Set a variable to the plot area of the chart.
    Set plPlotArea = chChart1.PlotArea

    ' Check to see if the rendered object is a gridline.
    If TypeName(chartObject) = "ChGridlines" Then

        ' The next four lines of code use the extents of
        ' the plot area to calculate the dimensions of the line
        ' to be drawn.
        lLeft = plPlotArea.Left
        lTop = plPlotArea.Top
        lRight = plPlotArea.Right
        lHeight = plPlotArea.Bottom - lTop
    End If
End Sub
```


```
' Determine the increment to use between gridlines.
' Change the divisor to adjust the increment.
lIncrement = lHeight / 10

' The next three lines of code set the properties of the
' line to be drawn.
drawObject.Line.DashStyle = chConstants.chLineRoundDot
drawObject.Line.Color = "Green"
drawObject.Line.Weight = chConstants.owcLineWeightMedium

For iCtr = 1 To 9

    ' Draw the line.
    drawObject.DrawLine lLeft, lTop + iCtr * lIncrement, _
                        lRight, lTop + iCtr * lIncrement

Next
End If
End Sub
```



▼ [Show All](#)

# BeforeScreenTip Event

► [BeforeScreenTip event as it applies to the \*\*ChartSpace\*\* object.](#)

Occurs before a ScreenTip is to be displayed.

**Private Sub *Object*\_BeforeScreenTip(ByVal *TipText* As ByRef, ByVal *ContextObject*)**

***Object*** The name of the **ChartSpace** object that you are trapping this event for.

***TipText*** Set the **Value** property of this object to the ScreenTip that you want to display. Set the **Value** property to "" to suppress the ScreenTip.

***ContextObject*** The object in the chart that triggered the ScreenTip.

► [BeforeScreenTip event as it applies to the \*\*PivotTable\*\* object.](#)

Occurs before a ScreenTip is to be displayed.

**Private Sub *Object*\_BeforeScreenTip(ByVal *ScreenTipText* As ByRef, ByVal *SourceObject*)**

***Object*** The name of the **PivotTable** object that you are trapping this event for.

***ScreenTip Text*** Set the **Value** property of this object to the screen tip that you want to display. Set the **Value** property to "" to suppress the ScreenTip.

***SourceObject*** The object in the chart that triggered the ScreenTip.



## Remarks

Use this event to customize ScreenTips displayed in a chart or PivotTable list.

--

# BeforeUpdate Event

-

Occurs when data is changed, but before the recordset is updated. Use this event to validate data before it is committed to the database.

**Private Sub *Object*\_BeforeUpdate(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

Set the **ReturnValue** property of the **DSCEventInfo** object to **False** to cancel the update.

## Example

This example cancels the updating of the recordset when the user enters a value greater than 0 for the UnitsOnOrder field when the UnitsInStock field is greater than 100.

```
Sub MSODSC_BeforeUpdate(DSCEventInfo)

    Dim txtUnitsOnOrder
    Dim txtUnitsInStock

    ' Set a variable to the text box that contains the value
    ' for the UnitsOnOrder field.
    Set txtUnitsOnOrder = DSCEventInfo.Section.HTMLContainer _
        .Children("UnitsOnOrder")

    ' Set a variable to the text box that contains the value
    ' for the UnitsInStock field.
    Set txtUnitsInStock = DSCEventInfo.Section.HTMLContainer _
        .Children("UnitsInStock")

    ' Check the value of the UnitsOnOrder Field.
    If CLng(txtUnitsOnOrder.Value) > 0 then

        ' Check the value of the UnitsInStock Field.
        If CLng(txtUnitsInStock.Value) > 100 then

            ' Display a message to the user.
            MsgBox "Don't reorder the part until fewer than 100 are in

            ' Cancel the update.
            DSCEventInfo.ReturnValue = False

        End If
    End If

End Sub
```

---



## ButtonClick Event

Occurs whenever the user clicks a navigation button.

```
Private Sub RecordNavigationControl_ButtonClick(NavButton As NavButtonEnum)
```

***NavButton*** Specifies the button that, when clicked, triggers the ButtonClick event. Can be one of the **NavButtonEnum** constants.

**Remarks**

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

# Change Event

-

Occurs whenever data in one or more cells changes. Both edits and copy-and-paste operations cause this event to occur.

**Private Sub Range\_Change( )**



## Remarks

This event occurs after the [EndEdit](#) event; at this point, the data has already been changed and the change cannot be canceled.

This event requires the WithEvents keyword, so it cannot be used with VBScript or JScript.

## Example

The following example updates a label control on a Visual Basic form when the value in cell A1 of Sheet1 in Spreadsheet1 changes.

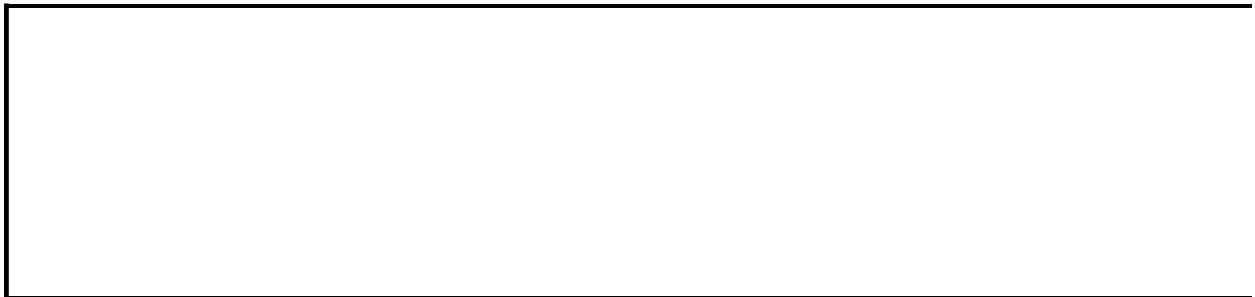
```
Dim WithEvents rngRange1 As Range

Private Sub Form_Load()

    ' Set a variable to the range for which you want to capture
    ' the Change event.
    Set rngRange1 = Spreadsheet1.Worksheets("Sheet1").Range("A1")

End Sub

Private Sub rngRange1_Change()
    ' Change the caption of Label1 to the current value
    ' of cell A1.
    Label1.Caption = rngRange1.Value
End Sub
```



# Click Event

-

Occurs whenever the user clicks the specified control.

**Private Sub** *object\_Click*( )

*object* A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

# CommandBeforeExecute Event

-

Occurs before a command is executed. Use this event when you want to impose certain restrictions before a command is executed.

**Private Sub *object*\_CommandBeforeExecute (ByVal *Command* As Variant, ByVal *Cancel* As ByRef)**

*object* A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

***Command*** Required. The command that has been executed.

***Cancel*** Required. Set the **Value** property of this object to **True** to cancel the command.

## Remarks

The **OCCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each of the Microsoft Office Web Components.

## Example

This example refreshes PivotTable1 when the export command is invoked so that the latest data is exported to Microsoft Excel.

```
Sub PivotTable1_CommandBeforeExecute(Command, Cancel)
```

```
    Dim ptConstants
```

```
    Set ptConstants = PivotTable1.Constants
```

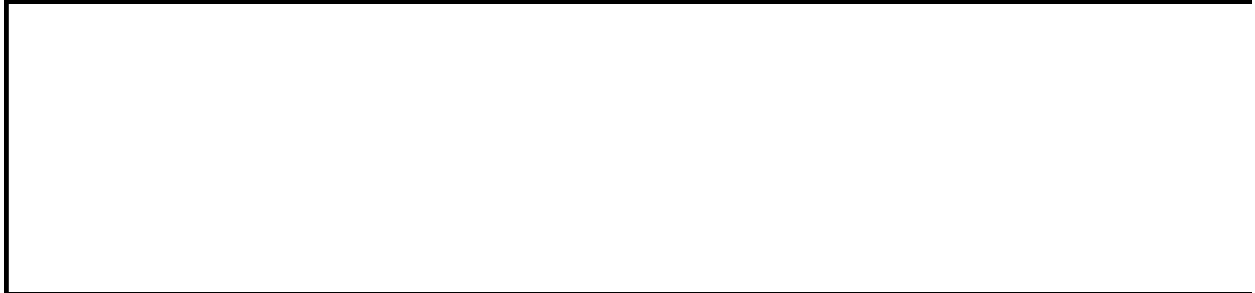
```
    ' Check to see if the Export command  
    ' has been invoked.
```

```
    If Command = ptConstants.plCommandExport Then
```

```
        ' Refresh the PivotTable list.  
        PivotTable1.Refresh
```

```
    End If
```

```
End Sub
```



# CommandChecked Event

-

Occurs when the the specified Microsoft Office Web Component determines whether a command is checked.

**Private Sub *object*\_CommandChecked (ByVal *Command* As Variant, ByVal *Checked* As ByRef)**

*object* A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

***Command*** Required. The command that has been executed.

***Checked*** Required. Set the **Value** property of this object to **True** to uncheck the command.



## Remarks

The **OCCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each Web component.

# CommandEnabled Event

-  
Occurs when the the specified Microsoft Office Web Component command is enabled.

**Private Sub *object*\_CommandEnabled (ByVal *Command* As Variant, ByVal *Enabled* As ByRef)**

*object* A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

***Command*** Required. The command that has been executed.

***Enabled*** Required. Set the **Value** property of this object to **True** to disable the command.

## Remarks

The **OCCommandID**, **ChartCommandIDEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each Web component.

# CommandExecute Event

-

Occurs after a command is executed. Use this event when you want to execute a set of commands after a particular command is executed.

**Private Sub** *object\_CommandExecute* (**ByVal** *Command* As Variant, **ByVal** *Succeeded* As Boolean)

*object* A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

**Command** The command that has been executed.

**Succeeded** Returns **True** if the command succeeded.

## Remarks

The **OCCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each of the Office Web Components.

## Example

This example writes the current date and time to a HTML text box control every time that PivotTable1 is refreshed.

```
Sub PivotTable1_CommandExecute(Command, Succeeded)

    Dim ptConstants

    Set ptConstants = PivotTable1.Constants

    ' Check to see if the PivotTable list has been refreshed.
    If Command = ptConstants.plCommandRefresh Then

        ' Write the current data and time to the text box.
        TextBox.Value = "PivotTable Last Refreshed on " & _
            Date & " at " & Time

    End If

End Sub
```

# CommandTipText Event

-  
Occurs when the the specified Microsoft Office Web Component queries a command's ScreenTip text.

**Private Sub *object*\_CommandTipText (ByVal *Command* As Variant, ByVal *Caption* As ByRef)**

*object* A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.

***Command*** The command that has been executed.

***Caption*** The **Value** property of this object contains the ScreenTip text for the command.

## Remarks

The **OCCommandId**, **ChartCommandIdEnum**, **PivotCommandId**, and **SpreadsheetCommandId** constants contain lists of the supported commands for each Web component.



# Current Event

-  
Occurs when a record becomes the current record.

**Private Sub *Object\_Current*(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

---

---

▼ [Show All](#)

# DataChange Event

-

Occurs when certain properties are changed or when certain methods are executed. See the [PivotDataReasonEnum](#) constant for more information about the properties and methods that can trigger this event.

PivotDataReasonEnum can be one of these PivotDataReasonEnum constants.

**plDataReasonAdhocFieldAdded**  
**plDataReasonAdhocFieldDeleted**  
**plDataReasonAdhocMemberChanged**  
**plDataReasonAllIncludeExcludeChange**  
**plDataReasonAllowDetailsChange**  
**plDataReasonAllowMultiFilterChange**  
**plDataReasonAlwaysIncludeInCubeChange**  
**plDataReasonCommandTextChange**  
**plDataReasonConnectionStringChange**  
**plDataReasonDataMemberChange**  
**plDataReasonDataSourceChange**  
**plDataReasonDisplayCalculatedMembersChange**  
**plDataReasonDisplayCellColorChange**  
**plDataReasonDisplayEmptyMembersChange**  
**plDataReasonExcludedMembersChange**  
**plDataReasonExpressionChange**  
**plDataReasonFieldNameChange**  
**plDataReasonFieldSetDeleted**  
**plDataReasonFieldSetNameChange**  
**plDataReasonFilterContextChange**  
**plDataReasonFilterCrossJoinsChange**  
**plDataReasonFilterFunctionChange**  
**plDataReasonFilterFunctionValueChange**

**plDataReasonFilterOnChange**  
**plDataReasonFilterOnScopeChange**  
**plDataReasonGroupEndChange**  
**plDataReasonGroupIntervalChange**  
**plDataReasonGroupOnChange**  
**plDataReasonGroupStartChange**  
**plDataReasonIncludedMembersChange**  
**plDataReasonInsertFieldSet**  
**plDataReasonInsertTotal**  
**plDataReasonIsFilteredChange**  
**plDataReasonIsIncludedChange**  
**plDataReasonMemberPropertyDisplayInChange**  
**plDataReasonMemberPropertyIsIncludedChange**  
**plDataReasonOrderedMembersChange**  
**plDataReasonRecordChanged**  
**plDataReasonRefreshDataSource**  
**plDataReasonRemoveFieldSet**  
**plDataReasonRemoveTotal**  
**plDataReasonSortDirectionChange**  
**plDataReasonSortOnChange**  
**plDataReasonSortOnScopeChange**  
**plDataReasonSubtotalsChange**  
**plDataReasonTotalAllMembersChange**  
**plDataReasonTotalDeleted**  
**plDataReasonTotalExpressionChange**  
**plDataReasonTotalFunctionChange**  
**plDataReasonTotalNameChange**  
**plDataReasonTotalSolveOrderChange**  
**plDataReasonUnknown**  
**plDataReasonUser**

**Private Sub *Object*\_DataChange(ByVal *Reason* As PivotDataReasonEnum)**

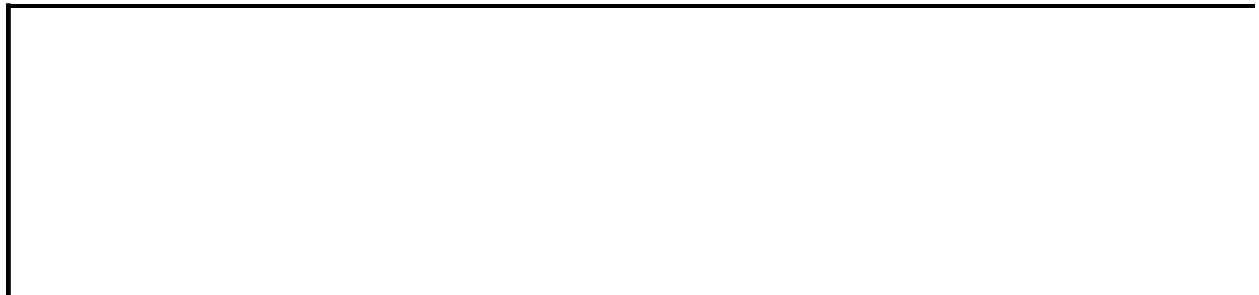
***Object*** The name of the **PivotTable** object that you are trapping this event for.

**Reason** Use the value of the [PivotDataReasonEnum](#) constant to determine the reason that this event was triggered.

PivotDataReasonEnum can be one of these PivotDataReasonEnum constants.

**plDataReasonAdhocFieldAdded**  
**plDataReasonAdhocFieldDeleted**  
**plDataReasonAdhocMemberChanged**  
**plDataReasonAllIncludeExcludeChange**  
**plDataReasonAllowDetailsChange**  
**plDataReasonAllowMultiFilterChange**  
**plDataReasonAlwaysIncludeInCubeChange**  
**plDataReasonCommandTextChange**  
**plDataReasonConnectionStringChange**  
**plDataReasonDataMemberChange**  
**plDataReasonDataSourceChange**  
**plDataReasonDisplayCalculatedMembersChange**  
**plDataReasonDisplayCellColorChange**  
**plDataReasonDisplayEmptyMembersChange**  
**plDataReasonExcludedMembersChange**  
**plDataReasonExpressionChange**  
**plDataReasonFieldNameChange**  
**plDataReasonFieldSetDeleted**  
**plDataReasonFieldSetNameChange**  
**plDataReasonFilterContextChange**  
**plDataReasonFilterCrossJoinsChange**  
**plDataReasonFilterFunctionChange**  
**plDataReasonFilterFunctionValueChange**  
**plDataReasonFilterOnChange**  
**plDataReasonFilterOnScopeChange**  
**plDataReasonGroupEndChange**  
**plDataReasonGroupIntervalChange**  
**plDataReasonGroupOnChange**  
**plDataReasonGroupStartChange**

**plDataReasonIncludedMembersChange**  
**plDataReasonInsertFieldSet**  
**plDataReasonInsertTotal**  
**plDataReasonIsFilteredChange**  
**plDataReasonIsIncludedChange**  
**plDataReasonMemberPropertyDisplayInChange**  
**plDataReasonMemberPropertyIsIncludedChange**  
**plDataReasonOrderedMembersChange**  
**plDataReasonRecordChanged**  
**plDataReasonRefreshDataSource**  
**plDataReasonRemoveFieldSet**  
**plDataReasonRemoveTotal**  
**plDataReasonSortDirectionChange**  
**plDataReasonSortOnChange**  
**plDataReasonSortOnScopeChange**  
**plDataReasonSubtotalsChange**  
**plDataReasonTotalAllMembersChange**  
**plDataReasonTotalDeleted**  
**plDataReasonTotalExpressionChange**  
**plDataReasonTotalFunctionChange**  
**plDataReasonTotalNameChange**  
**plDataReasonTotalSolverOrderChange**  
**plDataReasonUnknown**  
**plDataReasonUser**



# DataError Event

-  
Occurs whenever a data error occurs.

**Private Sub *Object\_DataError*(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.



# DataPageComplete Event

-  
Occurs when the specified data access page finishes loading.

**Private Sub *Object*\_DataPageComplete(*DSCEventInfo* As DSCEVENTINFO)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.





# DataSetChange Event

-

Occurs whenever a chart workspace is data-bound and the data set changes—for example, when a filter operation takes place. This event also occurs when initial data is available from the data source.

**Private Sub *Object*\_DataSetChange( )**

***Object*** The name of the **ChartSpace** object that this event applies to.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

# DblClick Event

-  
Occurs whenever the user double-clicks the specified control.

**Private Sub** *object*\_DblClick( )

*object* A [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object.



# Dirty Event

-  
Occurs when the contents of a data access page are changed by the user.

**Private Sub *Object\_Dirty*(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Set the **ReturnValue** property of the **DSCEventInfo** object to **False** to restore the previous value.

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

This event fires before the BeforeUpdate event.

# EndEdit Event

-

Occurs whenever the user switches from edit mode on the specified **Spreadsheet** Control or PivotTable list. You can use this event to validate data entry in a worksheet or in the detail area of a PivotTable list.

**Private Sub *Object\_EndEdit***(ByVal *Accept* As Boolean, ByVal *FinalValue* As ByRef, ByVal *Cancel* As ByRef, ByVal *ErrorDescription* As ByRef)

***Object*** A **PivotTable** or **Spreadsheet** object.

***Accept*** Specifies whether or not the specified control is accepting the edit. If this argument is **False**, then the control is leaving edit mode because the user cancelled the edit. If this argument is **True**, then you can cancel the edit.

***FinalValue*** The **Value** property of this argument returns the value that is to be entered into the worksheet or PivotTable list.

***Cancel*** Set the **Value** property of this argument to **True** to cancel the edit and leave the user in edit mode.

***ErrorDescription*** Set the **Value** property of this argument to the text that you want to display to the user. The default text is "The new value was not accepted."

---

---

# Focus Event

-  
Occurs when a section in a data access page receives focus.

**Private Sub *Object\_Focus*(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

--



# Initialize Event

-  
Occurs when the Spreadsheet Component is loading, but before it is loaded completely.

**Private Sub *Object*\_Initialize()**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

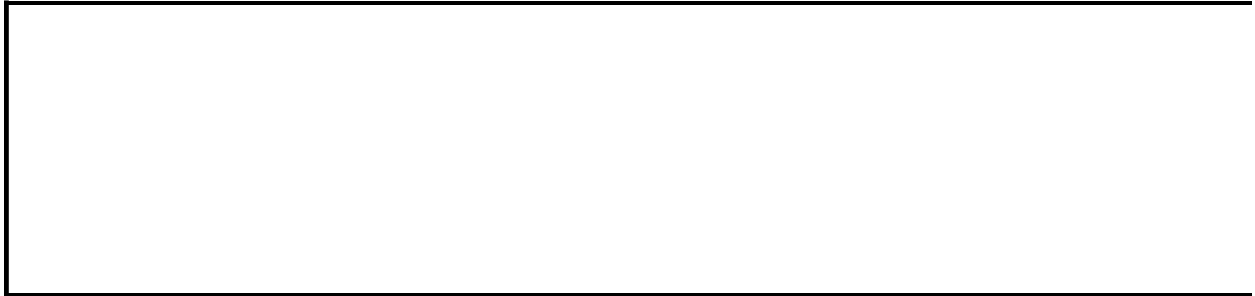
## Remarks

Use this event to initialize the settings for the spreadsheet.

## Example

This example uses the Initialize event to set the spreadsheet data from a file on the user's computer.

```
Sub Spreadsheet1_Initialize()  
    ' Load a CSV file into the spreadsheet.  
    Spreadsheet1.CSVURL = "Data.csv"  
End Sub
```



# KeyDown Event

-

Occurs whenever the user presses a key on the keyboard. If the user holds the key down, this event repeats itself at the key-repeat interval that has been set on the user's computer.

**Private Sub *Object*\_KeyDown(ByVal *KeyCode* As Long, ByVal *Shift* As Long)**

***Object*** The name of the **ChartSpace**, **PivotTable** or **Spreadsheet** object that you are trapping this event for.

***KeyCode*** A **Long** that represents the key code of the key that was pressed or released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, and 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

The sequence of keyboard-related events is:

1. BeforeKeyDown
2. KeyDown
3. BeforeKeyPress
4. KeyPress
5. BeforeKeyUp
6. KeyUp

--

# KeyPress Event

-  
Occurs whenever the user presses and releases a key on the keyboard.

**Private Sub *Object*\_KeyPress(ByVal *KeyAscii* As Long)**

***Object*** The name of the **ChartSpace**, **PivotTable** or **Spreadsheet** object that you are trapping this event for.

***KeyAscii*** A **Long** that represents the key code of the key that was pressed or released.

## Remarks

This event will not be called if the BeforeKeyDown event is cancelled.

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

The sequence of keyboard-related events is:

1. BeforeKeyDown
2. KeyDown
3. BeforeKeyPress
4. KeyPress
5. BeforeKeyUp
6. KeyUp

--

# KeyUp Event

-

Occurs whenever the user releases a key on the keyboard.

**Private Sub *Object*\_KeyUp(ByVal *KeyCode* As Long, ByVal *Shift* As Long)**

***Object*** The name of the **ChartSpace**, **PivotTable** or **Spreadsheet** object that you are trapping this event for.

***KeyCode*** A **Long** that represents the key code of the key that was pressed or released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, and 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.



## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

The sequence of keyboard-related events is:

1. BeforeKeyDown
2. KeyDown
3. BeforeKeyPress
4. KeyPress
5. BeforeKeyUp
6. KeyUp

--

# LoadCompleted Event

-  
Occurs when the Spreadsheet Component has completed loading.

**Private Sub *Object*\_LoadCompleted()**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

---

---

# MouseDown Event

-

Occurs whenever the user presses a mouse button while the pointer is positioned over the spreadsheet, PivotTable list, or the chart workspace.

**Private Sub *Object*\_MouseDown(ByVal *Button* As Long, ByVal *Shift* As Long, ByVal *x* As Long, ByVal *y* As Long)**

***Object*** The name of the **ChartSpace**, **PivotTable** or **Spreadsheet** object that you are trapping this event for.

***Button*** The mouse button that was released. Returns 1 if the primary mouse button was released, 2 if the secondary mouse button was released, or 4 if the middle mouse button was released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys when the event occurred. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, or 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.

***x*** The X coordinate of the mouse pointer.

***y*** The Y coordinate of the mouse pointer.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

# MouseMove Event

-  
Occurs whenever the user moves the mouse pointer over the PivotTable list or the chart workspace.

**Private Sub *Object*\_MouseMove**(ByVal *Button* As Long, ByVal *Shift* As Long, ByVal *x* As Long, ByVal *y* As Long)

***Object*** The name of the **ChartSpace** or **PivotTable** object that you are trapping this event for.

***Button*** The mouse button that was released. Returns 1 if the primary mouse button was released, 2 if the secondary mouse button was released, or 4 if the middle mouse button was released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys when the event occurred. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, or 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.

***x*** The X coordinate of the mouse pointer.

***y*** The Y coordinate of the mouse pointer.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

# MouseOut Event

-

Occurs whenever the user moves the mouse pointer out of a cell on a spreadsheet.

**Private Sub *Object*\_MouseOut(ByVal *Button* As Long, ByVal *Shift* As Long, ByVal *Target* As Range)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Button*** The mouse button that was released. Returns 1 if the primary mouse button was released, 2 if the secondary mouse button was released, or 4 if the middle mouse button was released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, or 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.

***Target*** A **Range** object that represents the cell or cells that the mouse pointer was moved out of.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).



# MouseOver Event

-

Occurs whenever the user moves the mouse pointer over a cell on the specified spreadsheet.

**Private Sub *Object*\_MouseOut(ByVal *Button* As Long, ByVal *Shift* As Long, ByVal *Target* As Range)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Button*** The mouse button that was released. Returns 1 if the primary mouse button was released, 2 if the secondary mouse button was released, or 4 if the middle mouse button was released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, or 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.

***Target*** A **Range** object that represents the cell or cells that the mouse pointer was moved over.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

# MouseUp Event

-

Occurs whenever the user releases a mouse button while the pointer is positioned over the spreadsheet, PivotTable list, or the chart workspace.

**Private Sub *Object*\_MouseUp(ByVal *Button* As Long, ByVal *Shift* As Long, ByVal *x* As Long, ByVal *y* As Long)**

***Object*** The name of the **ChartSpace**, **PivotTable** or **Spreadsheet** object that you are trapping this event for.

***Button*** The mouse button that was released. Returns 1 if the primary mouse button was released, 2 if the secondary mouse button was released, 4 if the middle mouse button was released.

***Shift*** The state of the SHIFT, CTRL, and ALT keys when the event occurred. Returns 1 if the SHIFT key was pressed, 2 if the CTRL key was pressed, and 4 if the ALT key was pressed. Returns 0 if neither the SHIFT, CTRL, nor ALT keys were pressed.

***x*** The x coordinate of the mouse pointer.

***y*** The y coordinate of the mouse pointer.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

# MouseWheel Event

-  
Occurs when the user rotates the mouse wheel on a mouse device that has a wheel.

**Private Sub *Object*\_MouseWheel (ByVal *Page* As Boolean, ByVal *Count* As Long)**

***Object*** The name of the **ChartSpace**, **PivotTable**, or **Spreadsheet** object that you are trapping this event for.

***Page*** Returns **True** if the page was changed.

***Count*** The number of lines that were scrolled.

---

# OnConnect Event

-

Occurs when the PivotTable list connects to a data source.

**Private Sub *Object*\_OnConnect()**

***Object*** The name of the **PivotTable** object that you are trapping this event for.

---

---

# OnDisconnect Event

-

Occurs when the PivotTable list is explicitly disconnected from a data source, or the connection is changed to a different data source.

**Private Sub *Object*\_OnDisconnect()**

***Object*** The name of the **PivotTable** object that you are trapping this event for.

--

▼ [Show All](#)



# PivotTableChange Event

-  
Occurs whenever a PivotTable list field, field set, or total is added or deleted.

**Private Sub PivotTable\_PivotTableChange(*Reason* As  
PivotTableReasonEnum)**

***Reason*** Specifies how the PivotTable list changed. Can be one of the  
[\*\*PivotTableReasonEnum\*\*](#) constants.

**plPivotTableReasonTotalAdded**

**plPivotTableReasonFieldSetAdded**

**plPivotTableReasonFieldAdded**

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

--

# Query Event

-

Occurs whenever a PivotTable list query becomes necessary. The query may not occur immediately; it may be delayed until the new data is displayed.

**Private Sub PivotTable\_Query( )**

--

# RecordExit Event

-

Occurs when the user navigates to another record, refreshes the data access page, or closes the data access page.

**Private Sub *Object\_RecordExit*(*DSCEventInfo* As DSCEventInfo)**

***Object*** The name of the **DataSourceControl** object that this event applies to.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

This event occurs after the BeforeUpdate event, but before the record is changed.

In the case of a banded data access page, moving among child records for the same parent does not fire this event.

Setting the **ReturnValue** property of the **DSCEventInfo** object to **False** cancels this event and prevents the record from being changed.

Use the **DataPage** and **Section** properties of the **DSCEventInfo** object to determine the data access page, section, and recordset that was updated.

--

# RecordsetSaveProgress Event

-

Occurs repeatedly when the **ExportXML** method is called. Use this event to provide feedback to the user when a recordset is exported.

**Private Sub *Object*\_RecordsetSaveProgress(ByVal *DSCEventInfo* As DSCEv**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Use the **PercentComplete** property to determine the current progress of the export operation.

You cannot use this event to update the contents of the current HTML document.

## Example

This example uses the RecordsetSaveProgress event to update Microsoft Internet Explorer's status bar when the recordset contained by the **DataSourceControl** is saved.

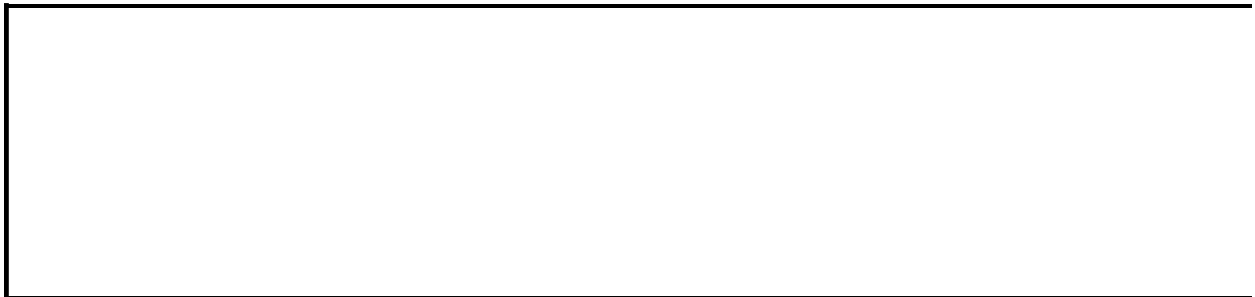
```
Sub MSODSC_RecordsetSaveProgress(DSCEventInfo)

    ' Update the status bar with the current
    ' completion percentage.
    Window.Status = DSCEventInfo.PercentComplete

    ' Check to see if the save has been completed.
    If DSCEventInfo.PercentComplete = 100 then

        ' Clear the status bar when the save is complete.
        Window.Status = ""
    End If

End Sub
```





# SelectionChange Event

-

Occurs whenever the user makes a new selection. The user cannot cancel this event.

**Private Sub *Object*\_SelectionChange( )**

***Object*** The name of the [ChartSpace](#), [PivotTable](#), or [Spreadsheet](#) object that this event applies to.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

You can use the [Selection](#) property to determine the object type of the current selection, as shown in the following example.

```
Private Sub PivotTable_SelectionChange()  
    If TypeName(PivotTable.Selection) = "PivotTotal" Then  
        'Handle selection of a total here  
    End If  
End Sub
```



# SelectionChanging Event

-

Occurs whenever the user moves the mouse pointer while selecting a range. This event does not occur when the user selects a range by using the keyboard. The user cannot cancel this event.

**Private Sub *Object* \_SelectionChanging(ByVal Range As Range)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Range*** A **Range** object that represents the range of cells that are being selected.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

--

# SheetActivate Event

-  
Occurs when a worksheet is activated.

**Private Sub *Object*\_SheetActivate(ByVal *Sh* As Worksheet)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Sh*** Required **Worksheet**. The worksheet that has been activated.

## Remarks

When a user changes worksheets, the SheetDeactivate event is called before the SheetActivate event.

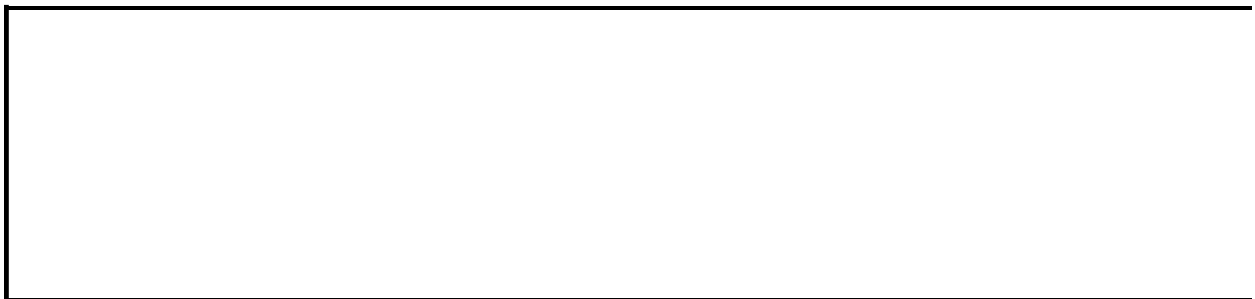
## Example

This example displays the name of the activated worksheet each time that a worksheet is activated in Spreadsheet1.

```
Sub Spreadsheet1_SheetActivate(Sh)
```

```
    MsgBox Sh.Name
```

```
End Sub
```



# SheetCalculate Event

-  
Occurs after any worksheet has been calculated.

**Private Sub *Object*\_SheetCalculate(ByVal *Sh* As Worksheet)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Sh*** A **Worksheet** object that represents the worksheet that was calculated.



## Example

This example uses the SheetCalculate event to monitor the status of the value in cell A5 in Sheet1 of Spreadsheet1.

```
Sub Spreadsheet1_SheetCalculate(Sh)

    Dim rngRangeToWatch

    ' Set a variable to the cell that you want to watch.
    Set rngRangeToWatch = Spreadsheet1.Worksheets("Sheet1").Range("A5")

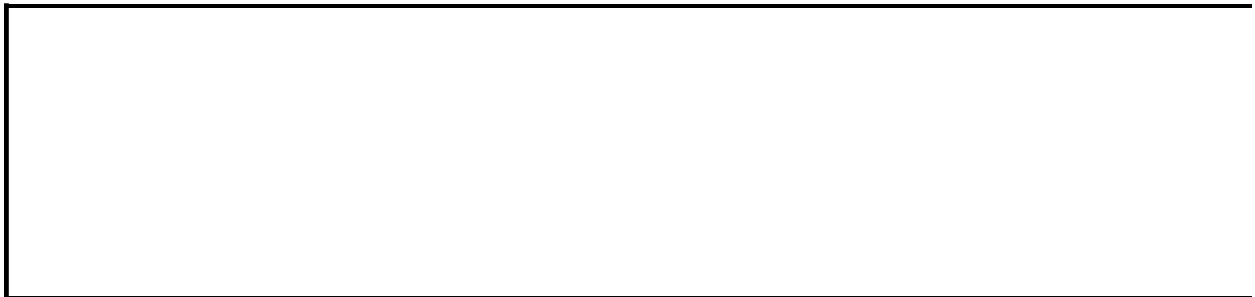
    ' If the calculated sheet is Sheet1...
    If Sh.Name = "Sheet1" Then

        ' ...and the value of the cell to watch is less than 10...
        If rngRangeToWatch.Value < 10 Then

            ' ... alert the user of the status.
            MsgBox "Inventory is less than 10. Reorder the part."
        End If

    End If

End Sub
```



# SheetChange Event

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

**Private Sub *Object*\_SheetChange(ByVal *Sh* As Object, ByVal *Target* As Range)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

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

***Target*** A **Range** object that represents the changed range.

## Example

This example illustrates how to use the SheetChange event to perform conditional formatting on cells A1:10 in Sheet1 of Spreadsheet1.

```
Sub Spreadsheet1_SheetChange(Sh, Target)
    Dim rngIntersect
    Dim rngCondFormat

    ' Set a variable to the range to be conditionally formatted. In
    ' case, the range is cells A1:A10 on Sheet1.
    Set rngCondFormat = Spreadsheet1.Worksheets("Sheet1").Range("A1:

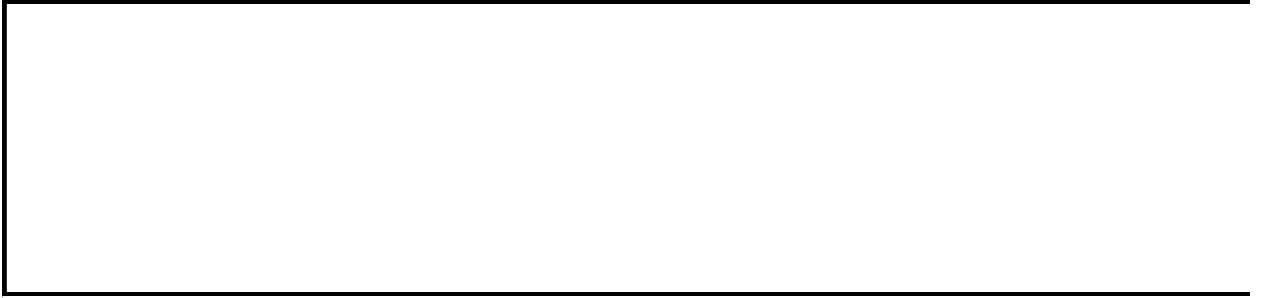
    ' Check to see if the change was made on Sheet1.
    If Sh.Name = "Sheet1" Then

        ' Set a variable to the intersection of the changed cell
        ' and the conditional formatting range.
        Set rngIntersect = Spreadsheet1.RectIntersect(Target, rngCon

        ' Check to see if the changed cell intersects with the
        ' conditional formatting range.
        If Not rngIntersect Is Nothing Then

            ' Format the target cell based on its value.
            Select Case Target.Value

                Case Is >= 25
                    Target.Font.Color = "Green"
                    Target.Font.Bold = True
                    Target.Font.Italic = False
                Case Is >= 10
                    Target.Font.Color = "Blue"
                    Target.Font.Bold = False
                    Target.Font.Italic = True
                Case Is < 10
                    Target.Font.Color = "Red"
                    Target.Font.Bold = True
                    Target.Font.Italic = False
            End Select
        End If
    End If
End Sub
```



# SheetDeactivate Event

-  
Occurs when a worksheet is deactivated.

**Private Sub *Object* \_SheetDeactivate(ByVal *Sh* As Worksheet)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Sh*** Required **Worksheet**. The worksheet that has been deactivated.

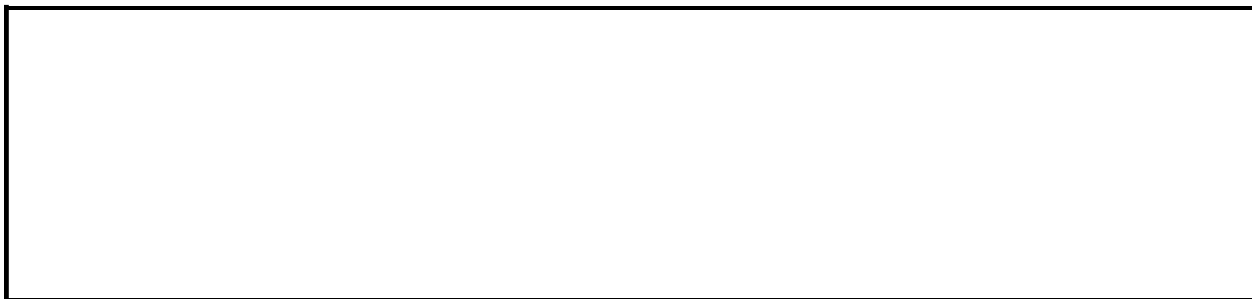
## Remarks

When a user changes worksheets, this event is called before the SheetActivate event.

## Example

This example displays the name of the deactivated worksheet each time that a worksheet is deactivated in Spreadsheet1.

```
Sub Spreadsheet1_SheetDeactivate(Sh)  
    MsgBox Sh.Name & " was just deactivated."  
End Sub
```



# SheetFollowHyperlink Event

-  
Occurs when a hyperlink is clicked.

**Private Sub *object*\_SheetFollowHyperlink(ByVal *Sh* As Worksheet, *Target* As Hyperlink)**

*object* Required. The name of a **Spreadsheet** object that you are trapping this event for.

***Sh*** Required. The worksheet that has been deactivated.

***Target*** Required. The hyperlink that has been clicked.



## Example

This example keeps a log of hyperlinks clicked in Spreadsheet1. The name of the sheet containing the hyperlink and the target address are written to Sheet3 each time that a hyperlink is clicked.

```
Sub Spreadsheet1_SheetFollowHyperlink(Sh, Target)

    Dim ssConstants
    Dim rngNewItem
    Dim shtListSheet

    Set ssConstants = Spreadsheet1.Constants

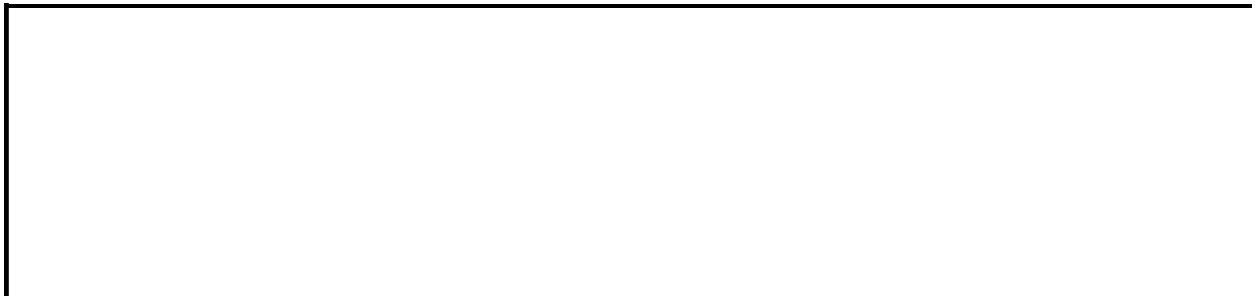
    ' Set a variable to Sheet3.
    Set shtListSheet = Spreadsheet1.ActiveWorkbook.Worksheets("Sheet

    ' Set a variable to the first available cell in column A of Shee
    Set rngNewItem = shtListSheet.Range("A262144").End(ssConstants.x

    ' Write the name of the sheet to Column A of Sheet3.
    rngNewItem.Value = Sh.Name

    ' Write the target address of the hyperlink to Column B of Sheet
    rngNewItem.Offset(0, 1).Value = Target.Address

End Sub
```



▼ [Show All](#)

# StartEdit Event

## ► [StartEdit event as it applies to the \*\*Spreadsheet\*\* object.](#)

Occurs whenever the user enters edit mode while the mouse pointer is in a cell.

**Private Sub *Object*\_StartEdit (ByVal *Selection* As Object, ByVal *InitialValue* As ByRef, ByVal *Cancel* As ByRef, ByVal *ErrorDescription* As ByRef)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Selection*** The currently selected range of cells.

***InitialValue*** Use the **Value** property of this object to return or set the initial value to be used when editing a cell.

***Cancel*** Set the **Value** property of this object to **True** to prevent the user from entering edit mode.

***ErrorDescription*** Set the **Value** property of this object to a message that you want to display to the user.

## ► [StartEdit Event as it applies to the \*\*PivotTable\*\* object.](#)

Occurs whenever the user enters edit mode in a detail cell.

**Private Sub *Object*\_StartEdit (ByVal *Selection* As Object, ByVal *ActiveObject* As Object, ByVal *InitialValue* As ByRef, ByVal *ArrowMode* As ByRef, ByVal *CaretPosition* As ByRef, ByVal *Cancel* As ByRef, ByVal *ErrorDescription* As ByRef)**

***Object*** The name of the **PivotTable** object that you are trapping this event for.

***Selection*** The currently selected range of cells.

**ActiveObject** The active object.

**InitialValue** Use the **Value** property of this object to return or set the initial value to be used when editing a cell.

**ArrowMode** Specifies how the left and right arrows function while the user is in edit mode. Set the **Value** property of this object to a [\*\*PivotArrowModeEnum\*\*](#) constant.

PivotArrowModeEnum can be one of these PivotArrowModeEnum constants.

**plArrowModeAccept** Accept the edit and move to the next cell.

**plArrowModeEdit** Move the insertion point left or right one position within the cell.

**CaretPosition** Specifies the position of the insertion point within the cell. Set the **Value** property of this object to a [\*\*PivotCaretPositionEnum\*\*](#) constant.

PivotCaretPositionEnum can be one of these PivotCaretPositionEnum constants.

**plCaretPositionAtEnd**

**plCaretPositionAtMouse**

**Cancel** Set the **Value** property of this object to **True** to prevent the user from entering edit mode.

**ErrorDescription** Set the **Value** property of this object to a message that you want to display to the user.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

--

# Undo Event

-

Occurs when the user clicks the **Undo** button on the navigation control, or the Dirty event is canceled. This event fires before the data is returned to its original values. Use this event to set the conditions under which the user is allowed to undo a change.

**Private Sub *Object*\_Undo(ByVal *DSCEventInfo* As DSCEventInfo)**

***Object*** A **DataSourceControl** object.

***DSCEventInfo*** The **DSCEventInfo** object that contains information about the event.

## Remarks

Set the **ReturnValue** property of the **DSCEventInfo** object to **False** to cancel the undo action.

You can use the **DataPage** and **Section** properties of the **DSCEventInfo** object to get more information about the page.

▼ [Show All](#)



# ViewChange Event

▸ [ViewChange event as it applies to the \*\*ChartSpace\*\* object.](#)

Occurs whenever the chart is redrawn.

**Private Sub *Object* \_ViewChange( )**

***Object*** The name of the **ChartSpace** object that you are trapping this event for.

▸ [ViewChange event as it applies to the \*\*PivotTable\*\* object.](#)

Occurs whenever the PivotTable list is redrawn.

**Private Sub *Object* \_ViewChange(ByVal *Reason* As PivotViewReasonEnum)**

***Object*** The name of the **PivotTable** object that you are trapping this event for.

***Reason*** The **PivotViewReasonEnum** constant that indicates how the view was changed.

▸ [ViewChange event as it applies to the \*\*Spreadsheet\*\* object.](#)

Occurs whenever the spreadsheet is redrawn.

**Private Sub *Object* \_ViewChange(ByVal *Target* As Range)**

***Object*** The name of the **Spreadsheet** object that you are trapping this event for.

***Target*** The currently visible range.

## Remarks

For information about using events with VBScript, see [Declaring and Using Event Procedures in VBScript](#).

--

▼ [Show All](#)

# ChartSpace Object Model





## Legend

Object and collection

Object only



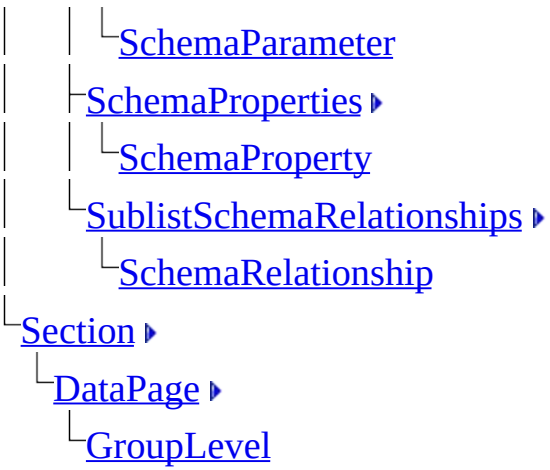
▼ [Show All](#)

# Data Source Control Object Model









## Legend

Object and collection

Object only



▼ [Show All](#)

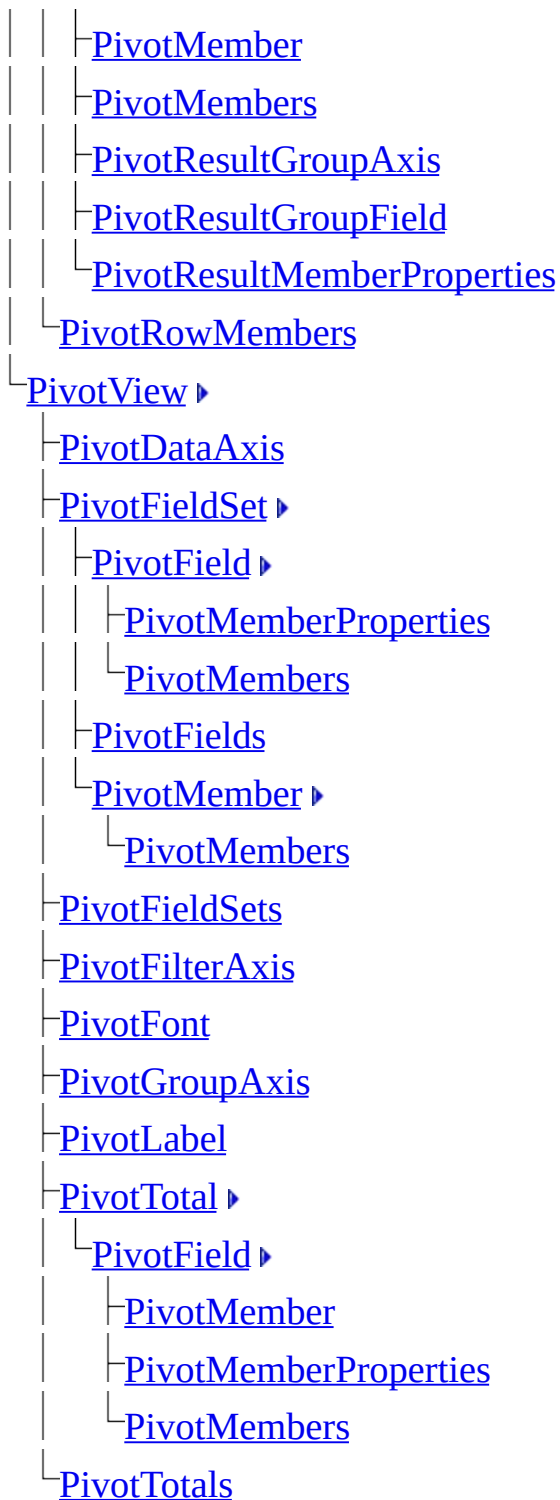
# PivotTable List Object Model





	<a href="#">PivotMemberProperty</a>
<a href="#">PivotColumnMember</a>	▶
<a href="#">PivotAxisMember</a>	
<a href="#">PivotAxisMembers</a>	
<a href="#">PivotField</a>	▶
<a href="#">PivotFieldSet</a>	
<a href="#">PivotFont</a>	
<a href="#">PivotMemberProperties</a>	
<a href="#">PivotTotal</a>	
<a href="#">PivotHyperlink</a>	
<a href="#">PivotMember</a>	
<a href="#">PivotMembers</a>	
<a href="#">PivotResultGroupAxis</a>	▶
<a href="#">PivotAxis</a>	
<a href="#">PivotFields</a>	
<a href="#">PivotResultGroupFields</a>	
<a href="#">PivotResultGroupField</a>	▶
<a href="#">PivotResultAxis</a>	▶
<a href="#">PivotAxis</a>	▶
<a href="#">PivotFieldSets</a>	
<a href="#">PivotLabel</a>	
<a href="#">PivotResultMemberProperties</a>	▶
<a href="#">PivotResultMemberProperty</a>	
<a href="#">PivotColumnMembers</a>	
<a href="#">PivotRange</a>	
<a href="#">PivotResultColumnAxis</a>	▶
<a href="#">PivotAxis</a>	
<a href="#">PivotAxisMember</a>	
<a href="#">PivotFields</a>	
<a href="#">PivotGroupAxis</a>	

	<a href="#">PivotResultGroupFields</a>
<a href="#">PivotResultDataAxis</a>	▶
	<a href="#">PivotAxis</a>
	<a href="#">PivotDataAxis</a>
	<a href="#">PivotFields</a>
	<a href="#">PivotTotals</a>
<a href="#">PivotResultFilterAxis</a>	▶
	<a href="#">PivotAxis</a>
	<a href="#">PivotFieldSets</a>
	<a href="#">PivotFilterAxis</a>
<a href="#">PivotResultLabel</a>	▶
	<a href="#">PivotLabel</a>
<a href="#">PivotResultPageAxis</a>	▶
	<a href="#">PivotAxis</a>
	<a href="#">PivotAxisMember</a>
	<a href="#">PivotFields</a>
	<a href="#">PivotGroupAxis</a>
	<a href="#">PivotPageMember</a>
	<a href="#">PivotResultGroupFields</a>
<a href="#">PivotResultRowAxis</a>	▶
	<a href="#">PivotAxis</a>
	<a href="#">PivotAxisMember</a>
	<a href="#">PivotFields</a>
	<a href="#">PivotGroupAxis</a>
	<a href="#">PivotResultGroupFields</a>
<a href="#">PivotRowMember</a>	▶
	<a href="#">PivotAxisMember</a>
	<a href="#">PivotAxisMembers</a>
	<a href="#">PivotField</a>
	<a href="#">PivotHyperlink</a>



## Legend

Object and collection

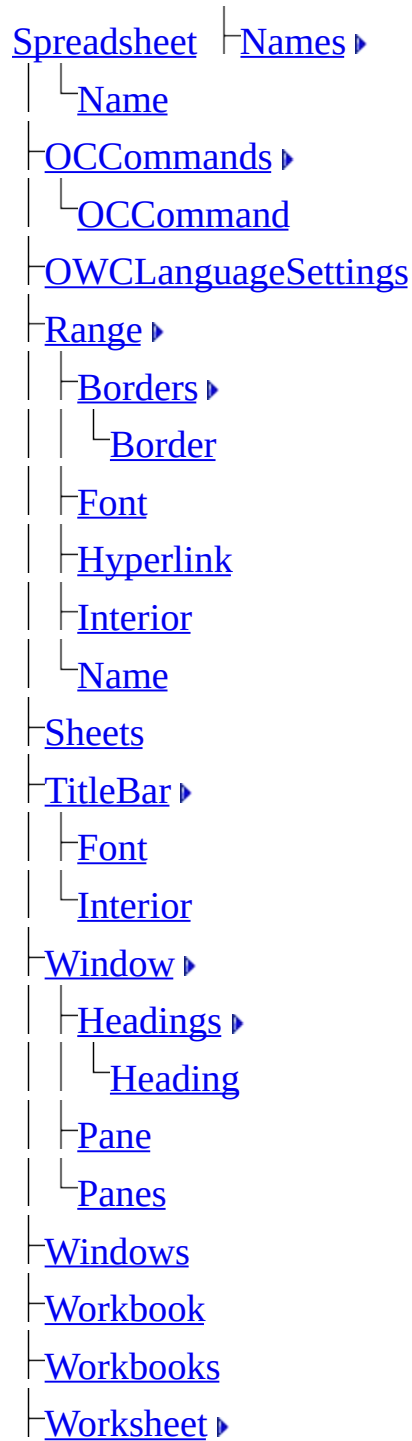
Object only

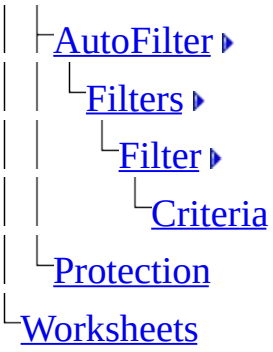


--

▼ [Show All](#)

# Spreadsheet Object Model





## Legend

Object and collection

Object only

--

# X Property

-

Returns a **Long** that represents the X-coordinate of the data point currently stored in the **Coordinate** object. Read-only.

*expression.x*

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

## Remarks

Use the **ValueToPoint** method to return the coordinates of a data point to a **Coordinate** object.

Use the **y** property to return the Y-coordinate of the data point currently stored in the **Coordinate** object.

## Example

This example changes the title of the first chart in Chartspace1 to the pixel coordinates of a data point in the first series of the chart.

```
Sub GetPixelCoordinates()  
  
    Dim chChart1  
    Dim lXPos  
    Dim lYPos  
    Dim coPointCoordinates  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set chChart1 = ChartSpace1.Charts(0)  
  
    ' Enable the title for the chart.  
    chChart1.HasTitle = True  
  
    ' Set a Coordinate object to the coordinates of a data point.  
    Set coPointCoordinates = chChart1.SeriesCollection(0).ValueToPoi  
  
    ' Set a variable to the X-coordinate.  
    lXPos = coPointCoordinates.x  
  
    ' Set a variable to the Y-coordinate.  
    lYPos = coPointCoordinates.y  
  
    ' Set the chart's titles to the pixel coordinates of the specifi  
    ' data point.  
    chChart1.Title.Caption = "X(" & lXPos & ") Y(" & lYPos & ")"  
  
End Sub
```



# y Property

-

Returns a **Long** that represents the Y coordinate of the data point currently stored in the **Coordinate** object. Read-only.

*expression.y*

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



## Remarks

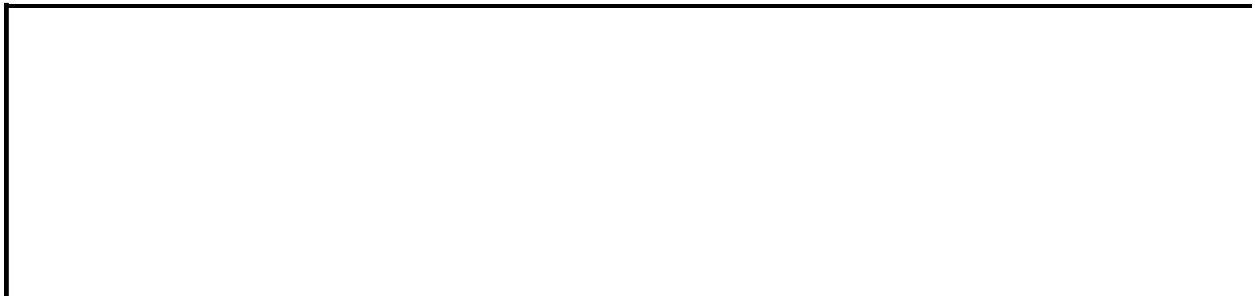
Use the **ValueToPoint** method to return the coordinates of a data point to a **Coordinate** object.

Use the **x** property to return the X coordinate of the data point currently stored in the **Coordinate** object.

## Example

This example changes the title of the first chart in Chartspace1 to the pixel coordinates of a data point in the first series of the chart.

```
Sub GetPixelCoordinates()  
  
    Dim chChart1  
    Dim lXPos  
    Dim lYPos  
    Dim coPointCoordinates  
  
    ' Set a variable to the first chart in Chartspace1.  
    Set chChart1 = ChartSpace1.Charts(0)  
  
    ' Enable the title for the chart.  
    chChart1.HasTitle = True  
  
    ' Set a Coordinate object to the coordinates of a data point.  
    Set coPointCoordinates = chChart1.SeriesCollection(0).ValueToPoi  
  
    ' Set a variable to the X-coordinate.  
    lXPos = coPointCoordinates.x  
  
    ' Set a variable to the Y-coordinate.  
    lYPos = coPointCoordinates.y  
  
    ' Set the chart's titles to the pixel coordinates of the specifi  
    ' data point.  
    chChart1.Title.Caption = "X(" & lXPos & ") Y(" & lYPos & ")"  
  
End Sub
```



# Returning an Object from a Collection

The [Item](#) property returns a single object from a collection. The following example sets the variable `thisChart` to a **ChChart** object that represents chart one.

```
Set thisChart = ChartWorkspace1.Charts.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 thisChart = ChartWorkspace1.Charts(1)
```

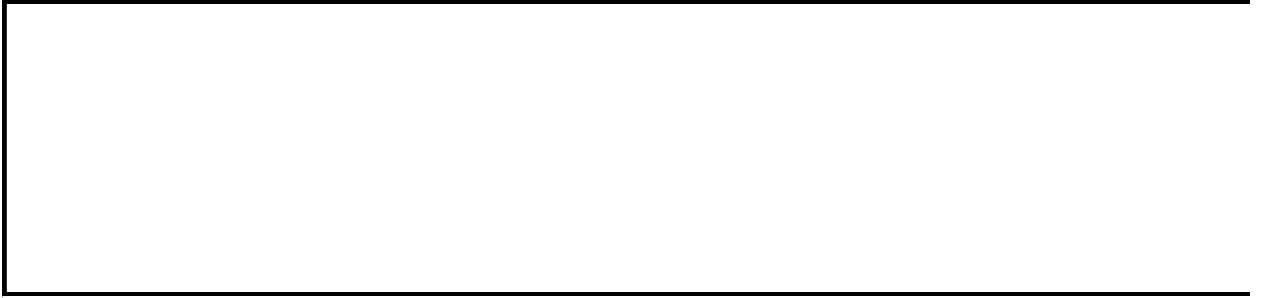
Some collections use an enumerated type with their **Item** property to return specific members of the collection. For example, the [ChAxes](#) collection uses the **ChartAxisPositionEnum** enumerated type, as shown in the following example.

```
Set chConstants = ChartSpace1.Constants  
Set valueAxis = ChartSpace1.Charts(0).Axes.Item(chConstants.chAxisPo  
Set categoryAxis = ChartSpace1.Charts(0).Axes.Item(chConstants.chAxi
```

Again, you can omit the **Item** keyword, as shown in the following example.

```
Set chConstants = ChartSpace1.Constants  
Set valueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisPositio  
Set categoryAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisPosi
```

For more information about a specific collection, see the Help topic for that collection.



# Using Named Constants in VBScript

You cannot use named constants in VBScript code. The following example works in Visual Basic but does not work in VBScript.

```
Set valueAxis = ChartSpace1.Charts(0).Axes(chAxisPositionLeft)
```

VBScript regards the named constant `chAxisPositionLeft` as just another uninitialized variable, so its value is 0 (zero). Because the actual value of `chAxisPositionLeft` is -3, this code does not work as expected in VBScript.

The [Constants](#) property returns an object that allows VBScript programmers to use named constants. This property applies to each of the top-level container objects (**ChartSpace**, **DataSourceControl**, **PivotTable**, and **Spreadsheet**). It returns an object that contains all of the named constants available in the Microsoft Office Web Components type library (no matter which object the **Constants** property is applied to, it always returns the complete set of named constants).

To use named constants in VBScript, you can set an object variable to the object returned by the **Constants** property and then use that object to qualify the named constants in your code, as shown in the following example.

```
Set chConstants = ChartSpace1.Constants  
Set valueAxis = ChartSpace1.Charts(0).Axes(chConstants.chAxisPositionLeft)
```

You can also use the **Constants** property directly in an expression, as shown in the following example.

```
Spreadsheet1.ActiveSheet.Export "sctest.xls", Spreadsheet1.Constants
```

**Note** You can use the **Constants** property in Visual Basic, but it is neither required nor recommended. Using the **Constants** property in containers where it is not required will cause your code to run significantly slower.



# Declaring and Using Event Procedures in VBScript

You declare event procedures in Visual Basic by using the **Private** and **ByVal** keywords and arguments with explicit type declarations, as shown in the following example.

```
Private Sub Spreadsheet1_MouseOver(ByVal Button As Long, ByVal Shift
```

This procedure declaration will not work in VBScript because VBScript does not use these keywords and because all arguments are passed as **Variant**. Instead, you declare event procedures in VBScript simply by using the event name and argument names, as shown in the following example.

```
Sub Spreadsheet1_MouseOver(Button, Shift, Target)
```

The argument names themselves are simply a convention in any container (you could use any argument names).

**Caution** Some script editors (including Microsoft Script Editor) do not fill in the argument list when they create an event procedure. To ensure that your event procedure runs correctly, consult the Object Browser or the appropriate event topic in Help, and fill in the argument list yourself.

