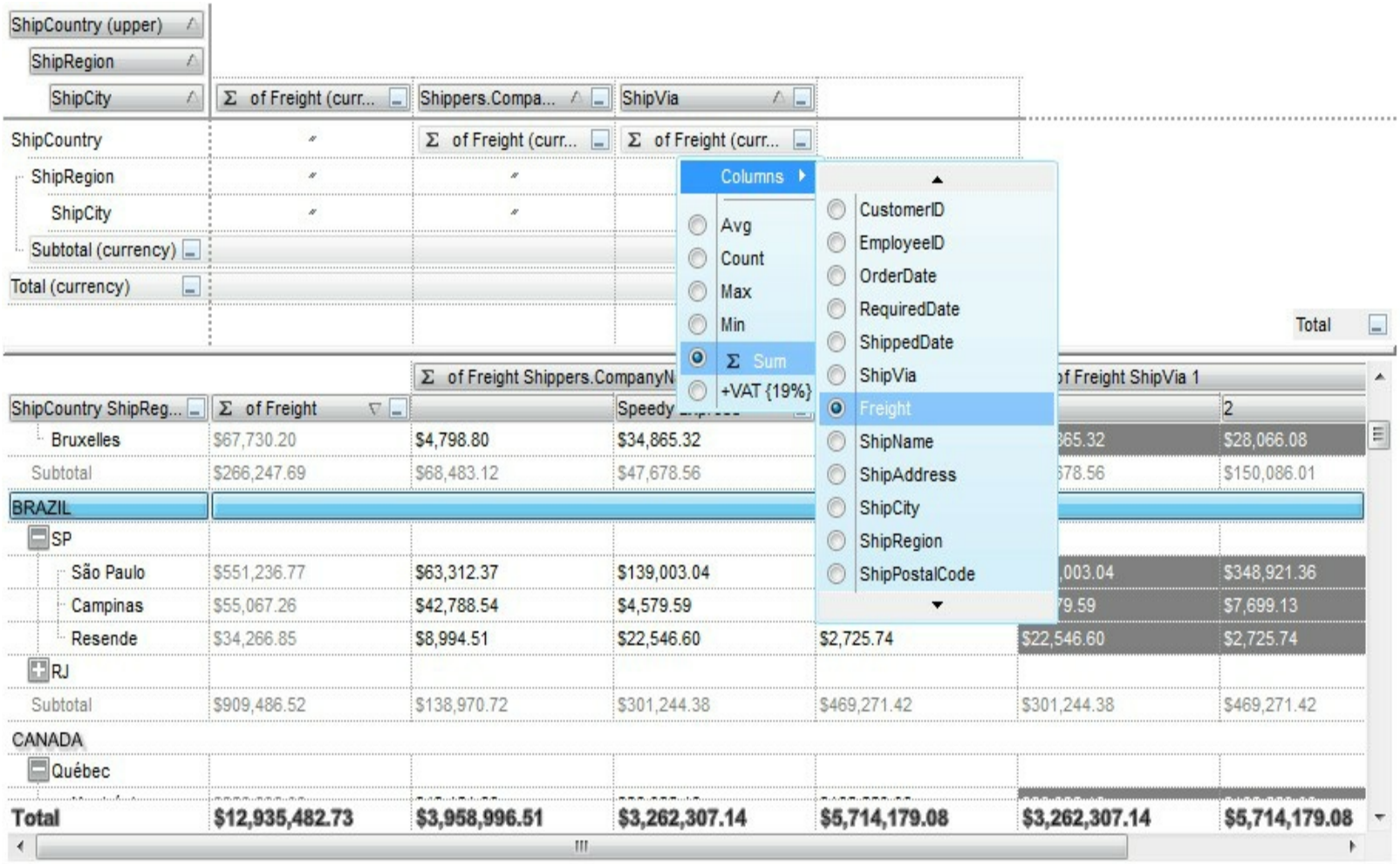
ExPivot

The Exontrol's eXPivot tool is our approach to provide data summarization, as a pivot table. A pivot-table can automatically sort, count, total or give the average of the data stored in one table or spreadsheet. The user sets up and changes the summary's structure by dragging and dropping fields graphically. The eXPivot component lets the user changes its visual appearance using skins, each one providing an additional visual experience that enhances viewing pleasure. Skins are relatively easy to build and put on any part of the control.

Features of eXPivot include:

- **Skinnable Interface** support (ability to apply a skin to any background part)
- Easy way to define the control's visual appearance in design mode, using XP-Theme elements or EBN objects
- **Print and Print-Preview** support, including Fit-To-Page
- Summarizes the data by **Drag and Drop**, or by code
- **Undo / Redo** support
- Ability to import data from **ADO/DAO** data-source, **TXT, XML** or by drag and drop from other sources
- Export data as **TXT, CSV, XML, EMF** or by drag and drop
- Ability to drag/filter the values of a column to the pivot bar, so the chart displays the associated function by value
- **SUM, AVG, COUNT, MIN or MAX** implementation, for numbers, string or date expressions
- Ability to add new aggregate functions, using built-in predefined operators
- **Total / Subtotal** rows support
- **Computed fields** support
- **Conditional Format** Support
- **PDF** (Portable Document Format), **BMP**, **JPG**, **GIF**, **PNG**, **TIFF**, **EMF** support (Ability to save/export the control's content to PDF, BMP, JPG, GIF, PNG, TIFF, EMF formats)
- Easy way to **save / restore** the control's **layout** ,column positions and size, sorting, filtering, scrolling and so on
- Interactive context menu to define the format to be displayed, including the colors, font, and so on
- **FilterBar / FilterPrompt** support
- Automatically update the Total/Subtotal rows, once a filter is applied, so the aggregate rows reflect the visible date only
- Ability to define the content of the columns / rows context menu
- Compact or Expanded mode support
- Built-in **HTML**, Decorative Text, Tooltip, Mouse Wheel, Icons, Pictures support and more ...



How to get support?

To keep your business applications running, you need support you can count on.

Here are few hints what to do when you're stuck on your programming:

- Check out the samples - they are here to provide some quick info on how things should be done
- Check out the how-to questions using the eXHelper tool
- Check out the help - includes documentation for each method, property or event
- Check out if you have the latest version, and if you don't have it send an update request here.
- Submit your problem(question) here.

Don't forget that you can contact our development team if you have ideas or requests for new components, by sending us an e-mail at support@exontrol.com (please include the name of the product in the subject, ex: exgrid) . We're sure our team of developers will try to find a way to make you happy - and us too, since we helped.

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

The AlignmentEnum type defines the alignment of the object. The AlignmentEnum type supports the following values.

Name	Value	Description
LeftAlignment	0	Left Alignment
CenterAlignment	1	Center Alignment
RightAlignment	2	Right Alignment

constants AllowGroupByEnum

The AllowGroupByEnum type specifies where the column can be dropped on the control's pivot bar. The AllowGroupBy property specifies where the column can be dropped on the control's pivot bar. The AllowGroupByEnum type supports the following values:

Name	Value	Description
exGroupByNone	0	The column can not be dropped the control's pivot bar.
exGroupByRows	1	The column can make part of the PivotRows property (group by section).
exGroupByColumnsHeader	2	The column can make part of the PivotColumns property (generate values to be shown on the control's columns header).
exGroupByColumnAggregate	4	The column can make part of the PivotColumns property (generate aggregate column)).
exGroupByAny	7	The column can be dropped anywhere on the control's pivot bar. The exGroupByAny is a combination of exGroupByRows, exGroupByColumnsHeader and exGroupByColumnAggregate flags.

constants AppearanceEnum

The AppearanceEnum enumeration is used to specify the control's appearance. Use the Appearance property to specify the control's appearance. The following values are supported:

Name	Value	Description
None2	0	No border
Flat	1	Flat border
Sunken	2	Sunken border
Raised	3	Raised border
Etched	4	Etched border
Bump	5	Bump border

constants AutoDragEnum

The AutoDragEnum type indicates what the control does when the user clicks and start dragging a row or an item. The AutoDrag property indicates the way the component supports the AutoDrag feature. The AutoDrag feature indicates what the control does when the user clicks an item and start dragging. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on. The SingleSel property specifies whether the control supports single or multiple selection. The drag and drop operation starts once the user clicks and moves the cursor up or down, if the SingleSel property is True, and if SingleSel property is False, the drag and drop starts once the user clicks, and waits for a short period of time. If SingleSel property is False, moving up or down the cursor selects the items by drag and drop.

The AutoDragEnum type supports the following values:

Name	Value	Description
exAutoDragNone	0	AutoDrag is disabled.
exAutoDragPosition	1	Reserved.
exAutoDragPositionKeepIndent	2	Reserved.
exAutoDragPositionAny	3	Reserved.
exAutoDragCopy	8	Drag and drop the selected items to a target application, and paste them as image or text. Pasting the data to the target application depends on the application. You can use the exAutoDragCopyText to specify that you want to paste as Text, or exAutoDragCopyImage as an image.
exAutoDragCopyText	9	Drag and drop the selected items to a target application, and paste them as text only. Ability to drag and drop the data as text, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop operation can start anywhere Click here to watch a movie on how exAutoDragCopyText works.
		Drag and drop the selected items to a target application, and paste them as image only. Ability to drag and drop the data as it looks, to your favorite

exAutoDragCopyImage	10	Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop operation can start anywhere Click here to watch a movie on how exAutoDragCopyImage works.
exAutoDragCopySnapShot	11	Drag and drop a snap shot of the current component. This option could be used to drag and drop the current snap shot of the control to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant.
exAutoDragScroll	16	The component is scrolled by clicking the item and dragging to a new position. This option can be used to allow user scroll the control's content with NO usage of the scroll bar, like on your IPhone. Ability to smoothly scroll the control's content. The feature is useful for touch screens or tables pc, so no need to click the scroll bar in order to scroll the control's content. Click here to watch a movie on how exAutoDragScroll works.
exAutoDragPositionOnShortTouch	256	exAutoDragPositionOnShortTouch. The object can be dragged from a position to another, but not outside of its group.
exAutoDragPositionKeepIndentOnShortTouch	512	exAutoDragPositionKeepIndentOnShortTouch. The object can be dragged to any position or to any parent, while the dragging object keeps its indentation.
exAutoDragPositionAnyOnShortTouch	768	exAutoDragPositionAnyOnShortTouch. The object can be dragged to any position or to any parent, with no restriction.
exAutoDragCopyOnShortTouch	2048	exAutoDragCopyOnShortTouch. Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnShortTouch	2304	exAutoDragCopyTextOnShortTouch. Drag and drop the selected objects to a target application, and paste them as text only.
		exAutoDragCopyImageOnShortTouch. Drag and

exAutoDragCopyImageOnShortTouch	2560	drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnShortTouch	2816	exAutoDragCopySnapShotOnShortTouch. Drag and drop a snap shot of the current component.
exAutoDragScrollOnShortTouch	4096	exAutoDragScrollOnShortTouch. The component is scrolled by clicking the object and dragging to a new position.
exAutoDragPositionOnRight	65536	exAutoDragPositionOnRight. The object can be dragged from a position to another, but not outside of its group.
exAutoDragPositionKeepIndentOnRight	131072	exAutoDragPositionKeepIndentOnRight. The object can be dragged to any position or to any parent, while the dragging object keeps its indentation.
exAutoDragPositionAnyOnRight	196608	exAutoDragPositionAnyOnRight. The object can be dragged to any position or to any parent, with no restriction.
exAutoDragCopyOnRight	524288	exAutoDragCopyOnRight. Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnRight	589824	exAutoDragCopyTextOnRight. Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnRight	655360	exAutoDragCopyImageOnRight. Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnRight	720896	exAutoDragCopySnapShotOnRight. Drag and drop a snap shot of the current component.
exAutoDragScrollOnRight	1048576	exAutoDragScrollOnRight. The component is scrolled by clicking the object and dragging to a new position.
exAutoDragPositionOnLongTouch	16777216	exAutoDragPositionOnLongTouch. The object can be dragged from a position to another, but not outside of its group.
exAutoDragPositionKeepIndentOnLongTouch	33554432	exAutoDragPositionKeepIndentOnLongTouch. The object can be dragged to any position or to any parent, while the dragging object keeps its indentation.
		exAutoDragPositionAnyOnLongTouch. The object

exAutoDragPositionAnyOnLongTouch	503316648	can be dragged to any position or to any parent, with no restriction.
exAutoDragCopyOnLongTouch	134217728	exAutoDragCopyOnLongTouch. Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnLongTouch	150994944	exAutoDragCopyTextOnLongTouch. Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnLongTouch	167772160	exAutoDragCopyImageOnLongTouch. Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnLongTouch	184549376	exAutoDragCopySnapShotOnLongTouch. Drag and drop a snap shot of the current component.
exAutoDragScrollOnLongTouch	268435456	exAutoDragScrollOnLongTouch. The component is scrolled by clicking the object and dragging to a new position.

constants BackgroundPartEnum

The BackgroundPartEnum type indicates parts in the control. Use the Background property to specify a background color or a visual appearance for specific parts in the control. A Color expression that indicates the background color for a specified part. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

If you refer a part of the scroll bar please notice the following:

- All BackgroundPartEnum expressions that starts with **exVS** changes a part in a vertical scroll bar
- All BackgroundPartEnum expressions that starts with **exHS** changes a part in the horizontal scroll bar
- Any BackgroundPartEnum expression that ends with **P** (and starts with exVS or exHS) specifies a part of the scrollbar when it is pressed.
- Any BackgroundPartEnum expression that ends with **D** (and starts with exVS or exHS) specifies a part of the scrollbar when it is disabled.
- Any BackgroundPartEnum expression that ends with **H** (and starts with exVS or exHS) specifies a part of the scrollbar when the cursor hovers it.
- Any BackgroundPartEnum expression that ends with no **H**, **P** or **D** (and starts with exVS or exHS) specifies a part of the scrollbar on normal state.

Name	Value	Description
exHeaderFilterBarButton	0	Specifies the background color for the drop down filter bar button.
exFooterFilterBarButton	1	Specifies the background color for the closing button in the filter bar. Use the ClearFilter method to remove the filter from the control.
exCellButtonUp	2	Specifies the background color for the cell's button, when it is up.
exCellButtonDown	3	Specifies the background color for the cell's button, when it is down.
exDropDownButtonUp	4	Specifies the visual appearance for the drop down button, when it is up. Usually the editors with a drop down portion displays a drop down button.
		Specifies the visual appearance for the drop down

exDropDownButtonDown	5	button, when it is down. Usually the editors with a drop down portion displays a drop down button.
exButtonUp	6	Specifies the visual appearance for the button inside the editor, when it is up.
exButtonDown	7	Specifies the visual appearance for the button inside the editor, when it is down.
exDateHeader	8	Specifies the visual appearance for the header in a calendar control.
exDateTodayUp	9	Specifies the visual appearance for the today button in a calendar control, when it is up.
exDateTodayDown	10	Specifies the visual appearance for the today button in a calendar control, when it is down.
exDateScrollThumb	11	Specifies the visual appearance for the scrolling thumb in a calendar control.
exDateScrollRange	12	Specifies the visual appearance for the scrolling range in a calendar control.
exDateSeparatorBar	13	Specifies the visual appearance for the separator bar in a calendar control.
exDateSelect	14	Specifies the visual appearance for the selected date in a calendar control.
exSliderRange	15	Specifies the visual appearance for the slider's bar.
exSliderThumb	16	Specifies the visual appearance for the thumb of the slider.
exSelectInPlace	17	Specifies the visual appearance for the selection when a drop down editor is focused and closed.
exSplitBar	18	exSplitBar. Specifies the visual appearance for control's split bar.
exSelBackColorFilter	20	Specifies the visual appearance for the selection in the drop down filter window. The drop down filter window shows up when the user clicks the filter button in the column's header.
exSelForeColorFilter	21	Specifies the foreground color for the selection in the drop down filter window.
exSpinUpButtonUp	22	Specifies the visual appearance for the up spin button when it is not pressed.
exSpinUpButtonDown	23	Specifies the visual appearance for the up spin button when it is pressed.

exSpinDownButtonUp	24	Specifies the visual appearance for the down spin button when it is not pressed.
exSpinDownButtonDown	25	Specifies the visual appearance for the down spin button when it is pressed.
exBackColorFilter	26	Specifies the background color for the drop down filter window.
exForeColorFilter	27	Specifies the foreground color for the drop down filter window.
exSortBarLinkColor	28	Indicates the color or the visual appearance of the links between columns in the control's sort bar.
exCursorHoverColumn	32	Specifies the visual appearance for the column when the cursor hovers the column. By default, the exCursorHoverColumn property is zero, and it has no effect, so the visual appearance for the column is not changed when the cursor hovers the header.
exDragDropBefore	33	Reserved.
exDragDropAfter	34	Reserved.
exDragDropListTop	35	Reserved.
exDragDropListBottom	36	Reserved.
exDragDropForeColor	37	Reserved.
exDragDropListOver	38	Reserved.
exDragDropListBetween	39	Reserved.
exDragDropAlign	40	Reserved.
exHeaderFilterBarActive	41	Specifies the visual appearance of the drop down filter bar button, while filter is applied to the column.
exToolTipAppearance	64	Indicates the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ShowToolTip method to display a custom tooltip.
exToolTipBackColor	65	Specifies the tooltip's background color.

exToolTipForeColor	66	Specifies the tooltip's foreground color.
exColumnsFloatBackColor	87	Specifies the background color for the Columns float bar.
exColumnsFloatScrollBackColor	88	Specifies the background color for the scroll bars in the Columns float bar.
exColumnsFloatScrollPressBackColor	89	Specifies the background color for the scroll bars in the Columns float bar, while the scroll part is pressed.
exColumnsFloatScrollUp	90	Specifies the visual appearance of the up scroll bar.
exColumnsFloatScrollDown	91	Specifies the visual appearance of the down scroll bar.
exColumnsFloatAppearance	92	Specifies the visual appearance for the frame/borders of the Column's float bar
exColumnsFloatCaptionBackColor	93	Specifies the visual appearance for caption, if the Background(exColumnsFloatAppearance) property is specified.
exColumnsFloatCaptionForeColor	94	Specifies the foreground color for the caption, if the Background(exColumnsFloatAppearance) property is specified.
exColumnsFloatCloseButton	95	Specifies the visual appearance for the closing button, if the Background(exColumnsFloatAppearance) property is specified.
exPivotBarAppearance	96	Specifies the visual appearance for the frame/borders of the control's PivotBar.
exPivotBarBackColor	97	Specifies the background color for the PivotBar.
exPivotBarForeColor	98	Specifies the foreground color for the PivotBar.
exContextMenuAppearance	99	Specifies the visual appearance of the control's context menu.
exContextMenuBackColor	100	Specifies the solid background color for the control's context menu.
exContextMenuForeColor	101	Specifies the text foreground color for the control's context menu.
exContextMenuSelBackColor	102	Specifies the solid/EBN selection's background color in the control's context menu.
exContextMenuSelBorderColor	103	Specifies the solid color to show the selection in the control's context menu.

exContextMenuSelForeColor	104	Specifies the selection's text foreground color in the control's context menu.
exColumnsPositionSign	182	Specifies the visual appearance for the position sign between columns, when the user changes the position of the column by drag an drop.
exPivotBarAddNew	183	Specifies the visual appearance for the add new button (plus) in the control's pivot bar.
exPivotBarReload	184	Specifies the visual appearance for the reload button in the control's pivot bar.
exPivotBarTotal	185	Specifies the visual appearance of the Total button within the control's pivot bar.
exTreeLinesColor	186	Specifies the color to show the tree-lines (connecting lines from the parent to the children)
exVSUp	256	The up button in normal state.
exVSUpP	257	The up button when it is pressed.
exVSUpD	258	The up button when it is disabled.
exVSUpH	259	The up button when the cursor hovers it.
exVSThumb	260	The thumb part (exThumbPart) in normal state.
exVSThumbP	261	The thumb part (exThumbPart) when it is pressed.
exVSThumbD	262	The thumb part (exThumbPart) when it is disabled.
exVSThumbH	263	The thumb part (exThumbPart) when cursor hovers it.
exVSDown	264	The down button in normal state.
exVSDownP	265	The down button when it is pressed.
exVSDownD	266	The down button when it is disabled.
exVSDownH	267	The down button when the cursor hovers it.
exVSLower	268	The lower part (exLowerBackPart) in normal state.
exVSLowerP	269	The lower part (exLowerBackPart) when it is pressed.
exVSLowerD	270	The lower part (exLowerBackPart) when it is disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.

exVSUpper	272	The upper part (exUpperBackPart) in normal state.
exVSUpperP	273	The upper part (exUpperBackPart) when it is pressed.
exVSUpperD	274	The upper part (exUpperBackPart) when it is disabled.
exVSUpperH	275	The upper part (exUpperBackPart) when the cursor hovers it.
exVSBack	276	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exVSBackP	277	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exVSBackD	278	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exVSBackH	279	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exHSLeft	384	The left button in normal state.
exHSLeftP	385	The left button when it is pressed.
exHSLeftD	386	The left button when it is disabled.
exHSLeftH	387	The left button when the cursor hovers it.
exHSThumb	388	The thumb part (exThumbPart) in normal state.
exHSThumbP	389	The thumb part (exThumbPart) when it is pressed.
exHSThumbD	390	The thumb part (exThumbPart) when it is disabled.
exHSThumbH	391	The thumb part (exThumbPart) when the cursor hovers it.
exHSRight	392	The right button in normal state.
exHSRightP	393	The right button when it is pressed.
exHSRightD	394	The right button when it is disabled.
exHSRightH	395	The right button when the cursor hovers it.
exHSLower	396	The lower part (exLowerBackPart) in normal state.
exHSLowerP	397	The lower part (exLowerBackPart) when it is pressed.
exHSLowerD	398	The lower part (exLowerBackPart) when it is disabled.

exHSLowerH	399	The lower part (exLowerBackPart) when the cursor hovers it.
exHSUpper	400	The upper part (exUpperBackPart) in normal state.
exHSUpperP	401	The upper part (exUpperBackPart) when it is pressed.
exHSUpperD	402	The upper part (exUpperBackPart) when it is disabled.
exHSUpperH	403	The upper part (exUpperBackPart) when the cursor hovers it.
exHSBack	404	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exHSBackP	405	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exHSBackD	406	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exHSBackH	407	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exSBtn	324	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), in normal state.
exSBtnP	325	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is pressed.
exSBtnD	326	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is disabled.
exSBtnH	327	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when the cursor hovers it .
exScrollHoverAll	500	Enables or disables the hover-all feature. By default (Background(exScrollHoverAll) = 0), the left/top, right/bottom and thumb parts of the control' scrollbars are displayed in hover state while the cursor hovers any part of the scroll bar (hover-all feature). The hover-all feature is available on Windows 11 or greater, if only left/top, right/bottom, thumb, lower and upper-background parts of the scrollbar are visible, no custom visual-appearance is applied to any visible part. The hover-all feature is always on If Background(exScrollHoverAll) = -1. The Background(exScrollHoverAll) = 1 disables the hover-all feature.

exVSThumbExt	503	The thumb-extension part in normal state.
exVSThumbExtP	504	The thumb-extension part when it is pressed.
exVSThumbExtD	505	The thumb-extension part when it is disabled.
exVSThumbExtH	506	The thumb-extension when the cursor hovers it.
exHSThumbExt	507	The thumb-extension in normal state.
exHSThumbExtP	508	The thumb-extension when it is pressed.
exHSThumbExtD	509	The thumb-extension when it is disabled.
exHSThumbExtH	510	The thumb-extension when the cursor hovers it.
exScrollSizeGrip	511	Specifies the visual appearance of the control's size grip when both scrollbars are shown.

constants BackModeEnum

Specifies how the control displays the selection.

Name	Value	Description
exOpaque	0	The selection is opaque.
exTransparent	1	The selection is transparent.
exGrid	2	The control paints a grid selection.

constants CheckStateEnum

Specifies the states for a checkbox in the control.

Name	Value	Description
Unchecked	0	Specifies whether the cell is unchecked.
Checked	1	Specifies whether the cell is checked.
PartialChecked	2	Specifies whether the cell is partial-checked..

constants DescriptionTypeEnum

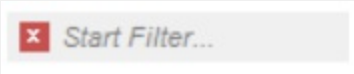
The DescriptionTypeEnum type defines captions that control displays. The Description property defines predefined captions being displayed on the control. The DescriptionTypeEnum type supports the following values:

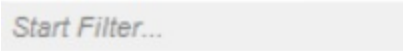
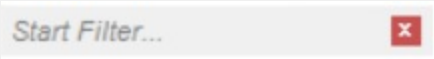
Name	Value	Description
exFilterBarAll	0	Defines the caption of '(All)' in the filter bar window.
exFilterBarBlanks	1	Defines the caption of '(Blanks)' in the filter bar window.
exFilterBarNonBlanks	2	Defines the caption of '(NonBlanks)' in the filter bar window.
exFilterBarFilterForCaption	3	Defines the caption of 'Filter For' in the filter bar window.
exFilterBarFilterTitle	4	Defines the title for the filter tooltip.
exFilterBarPatternFilterTitle	5	Defines the title for the filter pattern tooltip.
exFilterBarTooltip	6	Defines the tooltip for the filter window when it displays no pattern field.
exFilterBarPatternTooltip	7	Defines the tooltip for filter pattern window.
exFilterBarFilterForTooltip	8	Defines the tooltip for 'Filter For:' window.
exFilterBarIsBlank	9	Defines the caption of the function 'IsBlank' in the control's filter bar.
exFilterBarIsNonBlank	10	Defines the caption of the function 'not IsBlank' in the control's filter bar.
exFilterBarAnd	11	Customizes the ' and ' text in the control's filter bar when multiple columns are used to filter the items in the control.
exFilterBarDate	12	Specifies the 'Date:' caption being displayed in the drop down filter.
exFilterBarDateTo	13	Specifies the 'to' sequence being used to split the from date and to date in the Date field of the drop down filter window.
exFilterBarDateTooltip	14	Describes the tooltip that shows up when cursor is over the Date field.
exFilterBarDateTitle	15	Describes the title of the tooltip that shows up when the cursor is over the Date field.
		Specifies the caption for the 'Today' button in a date

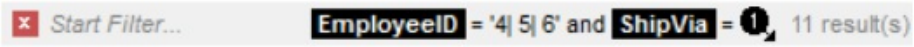
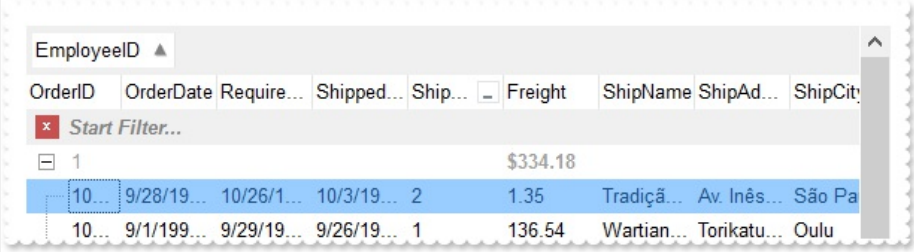
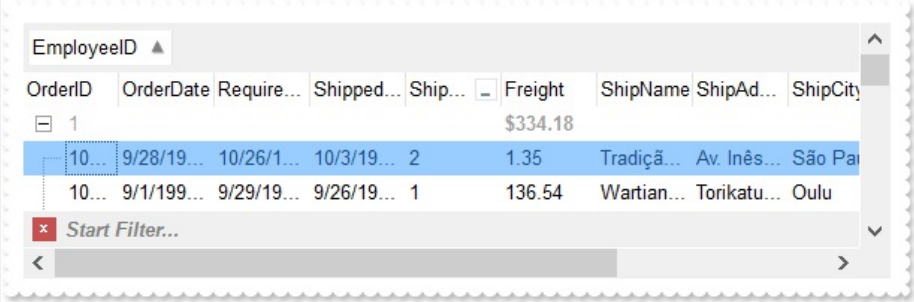
exFilterBarDateTodayCaption	16	filter window.
exFilterBarDateMonths	17	Specifies the name for months to be displayed in a date filter window.
exFilterBarDateWeekDays	18	Specifies the shortcut for the weekdays to be displayed in a date filter window.
exFilterBarChecked	19	Defines the caption of (Checked) in the filter bar window.
exFilterBarUnchecked	20	Defines the caption of (Unchecked) in the filter bar window.
exFilterBarIsChecked	21	Defines the caption of the 'IsChecked' function in the control's filter bar.
exFilterBarIsUnchecked	22	Defines the caption of the 'not IsChecked' function in the control's filter bar.
exFilterBarOr	23	Customizes the 'or' operator in the control's filter bar when multiple columns are used to filter the items in the control.
exFilterBarNot	24	Customizes the 'not' operator in the control's filter bar.
exFilterBarExclude	25	Specifies the 'Exclude' caption being displayed in the drop down filter.
exColumnsFloatBar	26	Specifies the caption to be shown on control's Columns float bar.

constants FilterBarVisibleEnum

The FilterBarVisibleEnum type defines the flags you can use on FilterBarPromptVisible property. The FilterBarCaption property defines the caption to be displayed on the control's filter bar. The FilterBarPromptVisible property , specifies how the control's filter bar is displayed and behave. The FilterBarVisibleEnum type includes several flags that can be combined together, as described bellow:

Name	Value	Description
exFilterBarHidden	0	No filter bar is shown while there is no filter applied. The control's filter bar is automatically displayed as soon a a filter is applied.
exFilterBarPromptVisible	1	The exFilterBarPromptVisible flag specifies that the control's filter bar displays the filter prompt. The exFilterBarPromptVisible, exFilterBarVisible, exFilterBarCaptionVisible flag , forces the control's filter-prompt, filter bar or filter bar description (even empty) to be shown. If missing, no filter prompt is displayed. The FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. 
exFilterBarVisible	2	The exFilterBarVisible flag forces the control's filter bar to be shown, no matter if any filter is applied. If missing, no filter bar is displayed while the control has no filter applied.  or combined with exFilterBarPromptVisible 
exFilterBarCaptionVisible	4	The exFilterBarVisible flag forces the control's filter bar to display the FilterBarCaption property. 

exFilterBarSingleLine	16	The exFilterBarVisible flag specifies that the caption on the control's filter bar id displayed on a single line. The exFilterBarSingleLine flag , specifies that the filter bar's caption is shown on a single line, so HTML tag or \r\n are not handled. By default, the control's filter description applies word wrapping. Can be combined to exFilterBarCompact to display a single-line filter bar. If missing, the caption on the control's filter bar is displayed on multiple lines. You can change the height of the control's filter bar using the FilterBarHeight property.
exFilterBarToggle	256	The exFilterBarToggle flag specifies that the user can close the control's filter bar (removes the control's filter) by clicking the close button of the filter bar or by pressing the CTRL + F, while the control's filter bar is visible. If no filter bar is displayed, the user can display the control's filter bar by pressing the CTRL + F key. While the control's filter bar is visible the user can navigate though the list or control's filter bar using the ALT + Up/Down keys. If missing, the control's filter bar is always shown if any of the following flags is present exFilterBarPromptVisible, exFilterBarVisible, exFilterBarCaptionVisible.
exFilterBarShowCloseIfRequired	512	The exFilterBarShowCloseIfRequired flag indicates that the close button of the control's filter bar is displayed only if the control has any currently filter applied. The Background(exFooterFilterBarButton) property on -1 hides permanently the close button of the control's filter bar. 
exFilterBarShowCloseOnRight	1024	The exFilterBarShowCloseOnRight flag specifies that the close button of the control's filter bar should be displayed on the right side. 
		The exFilterBarCompact flag compacts the control's filter bar, so the filter-prompt will be displayed to

exFilterBarCompact	2048	the left, while the control's filter bar caption will be displayed to the right. This flag has effect only if combined with the exFilterBarPromptVisible. This flag can be combined with the exFilterBarSingleLine flag, so all filter bar will be displayed compact and on a single line. 
exFilterBarTop	8192	The exFilterBarTop flag displays the filter-bar on top (between control's header and items section as shown:  By default, the filter-bar is shown aligned to the bottom (between items and horizontal-scroll bar) as shown: 

constants FilterIncludeEnum

The FilterIncludeEnum type defines the items to include when control's filter is applied. The FilterInclude property specifies the items being included, when the list is filtered. The FilterIncludeEnum type supports the following values:

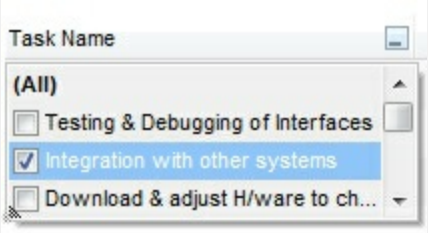
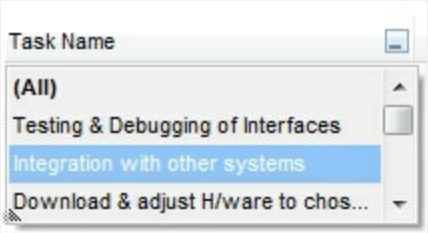
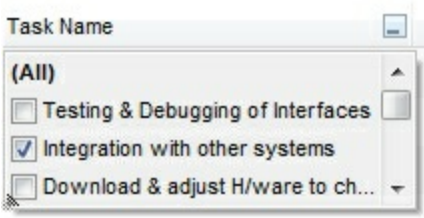
Name	Value	Description
exItemsWithoutChilds	0	Items (and parent-items) that match the filter are shown (no child-items are included)
exItemsWithChilds	1	Items (parent and child-items) that match the filter are shown
exRootsWithoutChilds	2	Only root-items (excludes child-items) that match the filter are displayed
exRootsWithChilds	3	Root-items (and child-items) that match the filter are displayed
exMatchingItemsOnly	4	Shows only the items that matches the filter (no parent or child-items are included)
exMatchIncludeParent	240	Specifies that the item matches the filter if any of its parent-item matches the filter. The exMatchIncludeParent flag can be combined with any other value.
exFilterIncludeDefault	241	Defines the default value for the FilterInclude property.

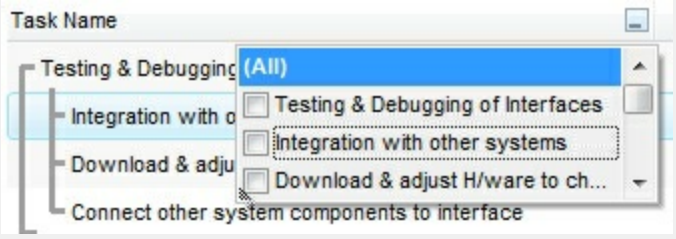
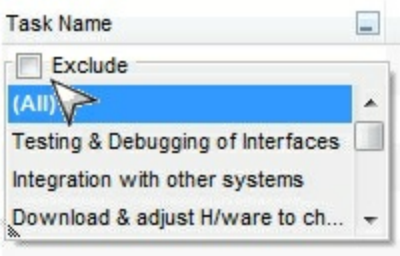
constants FilterListEnum

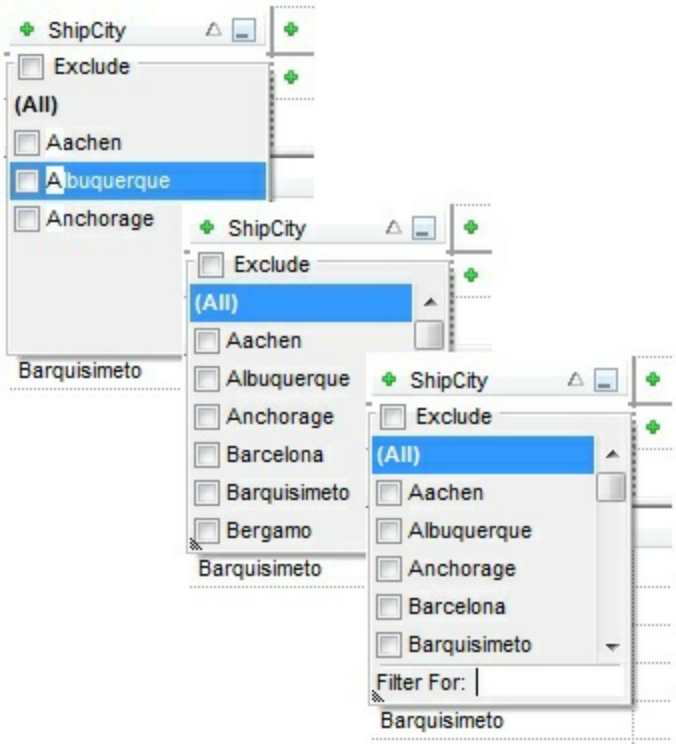
The FilterListEnum type specifies the type of items being included in the column's drop down list filter. The DisplayFilterList property specifies the items being included to the column's drop down filter-list, including other options for filtering. The SortType property of the Column object determines the type of the filtering box being displayed.

The FilterList can be a bit-combination of exAllItems, exVisibleItems or exNoItems with any other flags being described bellow:

Name	Value	Description
exAllItems	0	The filter's list includes all items in the column.
exVisibleItems	1	The filter's list includes only visible (filtered) items from the column. The visible items include child items of collapsed items.
exNoItems	2	The filter's list does not include any item from the column. Use this option if the drop down filter displays a calendar control for instance.
exLeafItems	3	The filter's list includes the leaf items only. A leaf item is an item with no child items.
exRootItems	4	The filter's list includes the root items only.
exSortItemsDesc	16	If the exSortItemsDesc flag is set the values in the drop down filter's list gets listed descending. If none of the exSortItemsAsc or exSortItemsDesc is present, the list is built as the items are displayed in the control.
exSortItemsAsc	32	If the exSortItemsAsc flag is set the values in the drop down filter's list gets listed ascending. If none of the exSortItemsAsc or exSortItemsDesc is present, the list is built as the items are displayed in the control.
exIncludeInnerCells	64	The exIncludeInnerCells flag specifies whether the inner cells values are included in the drop down filter's list.
exSingleSel	128	If this flag is present, the filter's list supports single selection. By default, (If missing), the user can select multiple items using the CTRL key. Use the exSingleSel property to prevent multiple items selection in the drop down filter list.
		The filter's list displays a check box for each

exShowCheckBox	256	included item. Clicking the checkbox, makes the item to be include din the filter. If this flag is present, the filter is closed once the user presses ENTER or clicks outside of the drop down filter window. By default, (this flag is missing), clicking an item closes the drop down filter, if the CTRL key is not pressed. This flag can be combined with exHideCheckSelect. The following screen shot shows the drop down filter **with** or **with no** exShowCheckBox flag:  or 
exHideCheckSelect	512	The selection background is not shown for checked items in the filter's list. This flag can be combined with exShowCheckBox. The following screen shot shows no selection background for the checked items: 
		This flag allows highlighting the focus cell value in the filter's list. The focus cell value is the cell's content at the moment the drop down filter window is shown. For instance, click an item so a new item is selected, and click the drop down filter button. A item being focused in the drop down filter list is the one you have in the control's selection. This flag has

exShowFocusItem	1024	effect also, if displaying a calendar control in the drop down filter list. The following screen shot shows the focused item in the filter's list (The Integration ... item in the background is the focused item, and the same is in the filter's list) : 
exShowPrevSelectOpaque	2048	By default, the previously selection in the drop down filter's list is shown using a semi-transparent color. Use this flag to show the previously selection using an opaque color. The exSelFilterForeColor and exSelFilterBackColor options defines the filter's list selection foreground and background colors.
exEnableToolTip	4096	This flag indicates whether the filter's tooltip is shown.
exShowExclude	8192	This flag indicates whether the Exclude option is shown in the drop down filter window. This option has effect also if the drop down filter window shows a calendar control. The following screen shot shows the Exclude field in the drop down filter window: 
exShowBlanks	16384	This flag indicates whether the (Blanks) and (NonBlanks) items are shown in the filter's list
		This flag indicates that no Filter For field is shown on the drop down filter window. Once the Filter For field is not shown, you can filter for items as soon as you type like shown in the following screen shot:

exHideFilterPattern	32768	
exFilterListDefault	9504	Defines the default value for the DisplayFilterList property. The exFilterListDefault value is a bit-OR combination of: exAllItems | exSortItemsAsc | exShowCheckBox | exShowFocusItem | exShowExclude

constants FilterPromptEnum

The FilterPromptEnum type specifies the type of prompt filtering. Use the FilterBarPromptType property to specify the type of filtering when using the prompt. The FilterBarPromptColumns specifies the list of columns to be used when filtering. The FilterBarPromptPattern property specifies the pattern for filtering. The pattern may contain one or more words being delimited by space characters.

The filter prompt feature supports the following values:

Name	Value	Description
exFilterPromptContainsAll	1	The list includes the items that contains all specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptContainsAny	2	The list includes the items that contains any of specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptStartWith	3	The list includes the items that starts with any specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptEndWith	4	The list includes the items that ends with any specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptPattern	16	The filter indicates a pattern that may include wild characters to be used to filter the items in the list. Can be combined with exFilterPromptCaseSensitive. The FilterBarPromptPattern property may include wild characters as follows: • '?' for any single character • '*' for zero or more occurrences of any character • '#' for any digit character

- ' ' space delimits the patterns inside the filter

exFilterPromptCaseSensitive	256	Filtering the list is case sensitive. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith, exFilterPromptEndWith or exFilterPromptPattern
exFilterPromptStartWords	4608	The list includes the items that starts with specified words, in any position. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.
exFilterPromptEndWords	8704	The list includes the items that ends with specified words, in any position. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.
exFilterPromptWords	12800	The filter indicates a list of words. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.

constants GridLinesEnum

Defines how the control paints the grid lines.

Name	Value	Description
exNoLines	0	The control displays no grid lines.
exAllLines	-1	The control displays vertical and horizontal grid lines.
exRowLines	-2	The control paints grid lines only for current rows.
exHLines	1	Only horizontal grid lines are shown.
exVLines	2	Only vertical grid lines are shown.

constants GridLinesStyleEnum

The GridLinesStyle type specifies the style to show the control's grid lines. The GridLineStyle property indicates the style of the gridlines being displayed in the view if the DrawGridLines property is not zero. The GridLinesStyle enumeration specifies the style for horizontal or/and vertical gridlines in the control.

Name	Value	Description
exGridLinesDot	0	The control's gridlines are shown as dotted.
exGridLinesHDot4	1	The horizontal control's gridlines are shown as dotted.
exGridLinesVDot4	2	The vertical control's gridlines are shown as dotted.
exGridLinesDot4	3	The control's gridlines are shown as solid.
exGridLinesHDash	4	The horizontal control's gridlines are shown as dashed.
exGridLinesVDash	8	The vertical control's gridlines are shown as dashed.
exGridLinesDash	12	The control's gridlines are shown as dashed.
exGridLinesHSolid	16	The horizontal control's gridlines are shown as solid.
exGridLinesVSolid	32	The vertical control's gridlines are shown as solid.
exGridLinesSolid	48	The control's gridlines are shown as solid.
exGridLinesGeometric	512	The control's gridlines are drawn using a geometric pen. The exGridLinesGeometric flag can be combined with any other flag. A geometric pen can have any width and can have any of the attributes of a brush, such as dithers and patterns. A cosmetic pen can only be a single pixel wide and must be a solid color, but cosmetic pens are generally faster than geometric pens. The width of a geometric pen is always specified in world units. The width of a cosmetic pen is always 1.

constants HierarchyLineEnum

Defines how the control paints the hierarchy lines.

Name	Value	Description
exNoLine	0	The control displays no lines when painting the hierarchy.
exDotLine	-1	The control uses a dotted line to paint the hierarchy.
exSolidLine	1	The control uses a solid line to paint the hierarchy.
exThinLine	2	The control uses a thin line to paint the hierarchy.

constants LayoutChangingEnum

The LayoutChangingEnum type specifies different operations in the control. The LayoutStartChanging event notifies your application once the user/control is about to being an operation. The LayoutEndChanging event notifies your application once the current operation ends. Your application can be notified for the following operations:

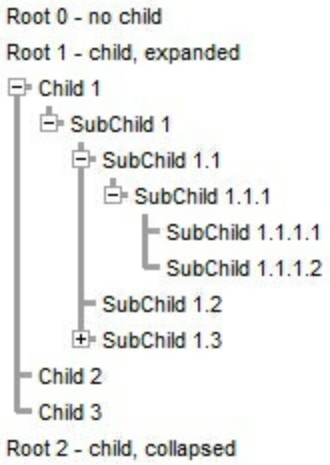
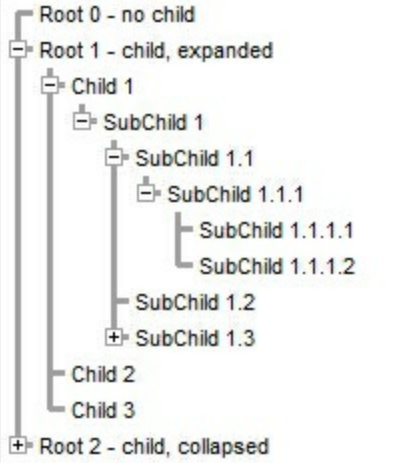
Name	Value	Description
exLayoutResizePanels	0	One of the panels has been resized. The PaneHeight property indicates the height of the panels. THe PaneMinHeight property specifies the minimum height of the specified panel. Include or exclude the exPivotBarSizable flag in the PivotBarVisible property, so you let the pivot bar to be sizable or not. Include or exclude the exPivotBarAutoFit flag in the PivotBarVisible property, so the pivot bar's height fits its content.
exLayoutPivotAutoHide	1	The pivot bar is auto shown or hidden based on the cursor position. Include or exclude the exPivotBarAutoHide flag in the PivotBarVisible property, to enable or disable the Auto-Hide feature of the control's pivot bar. If the exPivotBarAutoHide flag is included, the pivot bar is visible while the cursor hovers it, and it is hidden while the control is outside of the control's pivot bar. Include or exclude the exPivotBarSizable flag in the PivotBarVisible property, so you let the user resizes or not the pivot bar.
exPivotDataLayoutChange	2	The layout of data in the pivot bar is changing, such as dragging a column to pivot bar. This operation notifies your application once the user drag a column/aggregate to the pivot bar, or remove a column/aggregate from the pivot bar. In other words, this operation notifies once one of the following property is changed at runtime: • PivotRows property, specifies the list of data columns that build the rows in the pivot control (Group-By columns, first column). • PivotColumns property, specifies the list of data columns that build the columns in the pivot

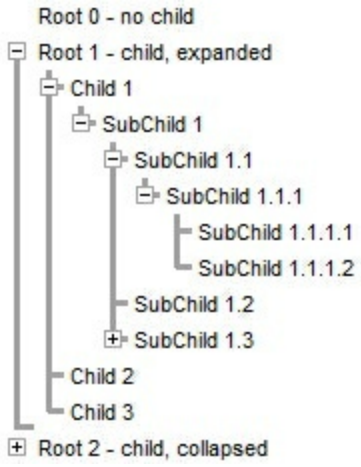
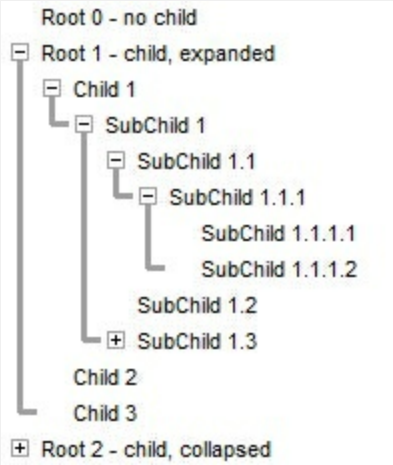
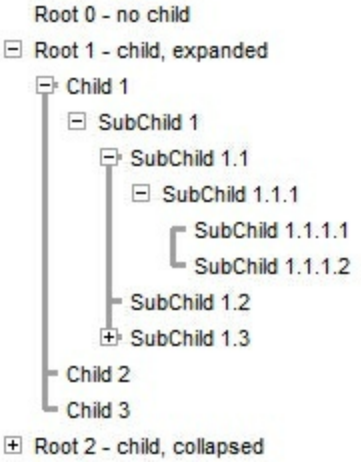
		control(the rest of columns, in the pivot control). • PivotTotals property, specifies the list of total/sub-totals/aggregate functions to be displayed in the pivot control.
exPivotDataLayoutExecute	3	The control executes the layout. The control arranges the data based on the current values of the following properties: • PivotRows property, specifies the list of data columns that build the rows in the pivot control (Group-By columns, first column). • PivotColumns property, specifies the list of data columns that build the columns in the pivot control(the rest of columns, in the pivot control). • PivotTotals property, specifies the list of total/sub-totals/aggregate functions to be displayed in the pivot control.
exPivotDataLayoutUndo	4	The user performs an Undo operation in the layout of the pivot's data. The Undo operation is performed once the user presses the CTRL+Z keys combination. Include or exclude the exPivotBarAllowUndoRedo flag in the PivotBarVisible property, to enable or disable the Undo/Redo feature of the control's pivot bar.
exPivotDataLayoutRedo	5	The user performs a Redo operation in the layout of the pivot's data. The Redo operation is performed once the user presses the CTRL+Y keys combination. Include or exclude the exPivotBarAllowUndoRedo flag in the PivotBarVisible property, to enable or disable the Undo/Redo feature of the control's pivot bar.
exPivotDataColumnFilterChange	6	The user changes the filter for a pivot column.
exPivotDataColumnSort	7	The user sorts a pivot column.

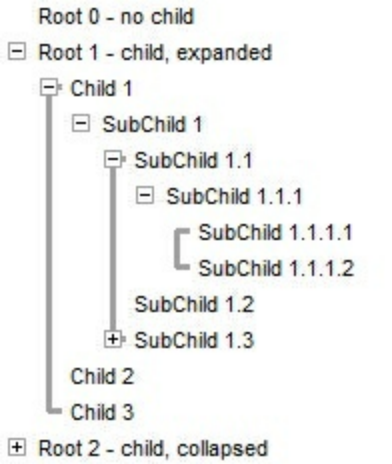
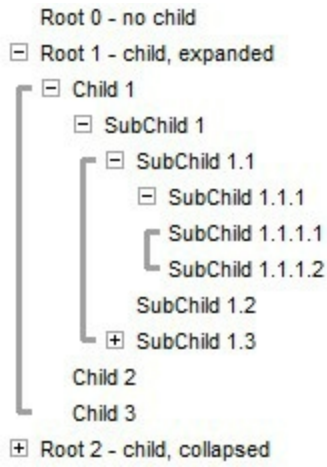
constants LinesAtRootEnum

Defines how the control displays the lines at root. The LinesAtRoot property defines the way the tree lines are shown. The HasLines property defines the type of the line to be shown. The HasButtons property defines the expand/collapse buttons for parent items.

The LinesAtRootEnum type support the following values:

Name	Value	Description
exNoLinesAtRoot	0	No lines at root items. 
exLinesAtRoot	-1	The control links the root items. 
		The control shows no links between roots, and divides them as being in the same group.

exGroupLinesAtRoot	1	
exGroupLines	2	The lines between root items are no shown, and the links show the items being included in the group. 
exGroupLinesInside	3	The lines between root items are no shown, and the links are shown between child only. 
		The lines between root items are no shown, and the links are shown for first and last visible child item.

exGroupLinesInsideLeaf	4		
exGroupLinesOutside	5	The lines between root items are no shown, and the links are shown for first and last visible child item. A parent item that contains flat child items only, does not indent the child part. By a flat child we mean an item that does not contain any child item. 	

constants IncludeExpandColumnEnum

The IncludeExpandColumnEnum type specifies whether the columns display its data when it contains child columns, and it is expanded. The IncludeExpandColumn property specifies whether the column itself is displayed in the list (header/chart), while it expanded (the column contains child columns). The IncludeExpandColumnEnum type supports the following values:

Name	Value	Description
exExcludeExpandColumn	0	The column itself is not displayed when it is expanded and it contains child columns (default).
exIncludeExpandColumn	1	The column itself is displayed, when it is expanded and it contains child columns. This flag can be combined with the exIncludeExpandColumnSingle flag.
exIncludeExpandColumnSingle	16	The column itself is displayed, even if it contains a single child column. This flag can be combined with the exIncludeExpandColumn flag.

constants OnFilterChangeEnum

The OnFilterChangeEnum type specifies the action the control performs once the user applies a filter to any of the columns. The OnFilterChangeEnum type supports the following values:

Name	Value	Description
exFilterNope	0	No action is performed once the user applies a filter to any of the control's columns.
exFilterHideTotals	1	The total/sub-total fields are hidden while the control displays filtered data.
exFilterUpdateTotals	3	(default) The total/sub-total fields are updated to reflect the visible rows, instead the entire data. For instance, you have a Grand Total field, which displays the total for all countries, and if the user applies a filter for only Spain and UK is shown, the Grand Total Field will display the total for the displayed countries only, so just for Spain + UK.

constants PictureDisplayEnum

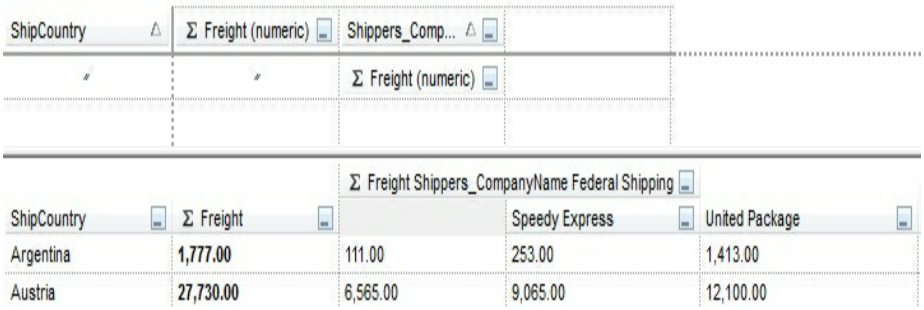
Specifies how a picture object is displayed.

Name	Value	Description
UpperLeft	0	Aligns the picture to the upper left corner.
UpperCenter	1	Centers the picture on the upper edge.
UpperRight	2	Aligns the picture to the upper right corner.
MiddleLeft	16	Aligns horizontally the picture on the left side, and centers the picture vertically.
MiddleCenter	17	Puts the picture on the center of the source.
MiddleRight	18	Aligns horizontally the picture on the right side, and centers the picture vertically.
LowerLeft	32	Aligns the picture to the lower left corner.
LowerCenter	33	Centers the picture on the lower edge.
LowerRight	34	Aligns the picture to the lower right corner.
Tile	48	Tiles the picture on the source.
Stretch	49	The picture is resized to fit the source.

constants PivotBarVisibleEnum

The PivotBarVisibleEnum type indicates the options/flags/properties of the control's pivot bar. The PivotBarVisible property indicates the options to be changed for the control's pivot bar. Any of the following values can be bit-OR combinate with any other flag. The PivotBarVisibleEnum type supports the following values:

Name	Value	Description
exPivotBarVisible	1	The pivot bar to show the columns is visible. Include or exclude this flag to show or hide the control's pivot bar. The PivotColumnsFloatBarVisible property indicates whether the control shows a floating panel to display the pivot columns that can be dropped to the control's pivot bar.
exPivotBarSizable	2	The pivot bar is sizable. Include or exclude this flag to allow or prevent resizing the pivot bar at runtime. If this flag is present, the user can resize the pivot bar by dragging the bottom side of the control. The resize cursor is shown when the pivot bar is resizable and cursor hovers the bottom side of the pivot bar. The PaneHeight property specifies the height of the top/bottom parts. The top panel is the control's pivot bar, while the bottom part is the control's list where the result goes. The PaneMinHeight property specifies the minimum height of the top/bottom parts.
exPivotBarFloat	4	The pivot bar is shown to a floating window. Include this flag, to display the control's pivot bar to a separate window (a floating panel).
exPivotBarAutoFit	8	Changes automatically the height of the pivot bar so all elements fits the client area. When this flag is present, the height of the pivot's bar is automatically adjusted so the entire content fits the header. The PaneHeight property specifies the height of control's panels. The PaneMinHeight property specifies the min height of control's panels.
exPivotBarShowTotals	16	Specifies if the control handles the Total/SubTotal fields. By default, the control displays a Total field in the pivot bar. Include or exclude this flag to show or hide the Total field in the pivot bar. In other words, you can use this option to allow or prevent using the

		total/sub-totals function in the control.
exPivotBarAutoHide	32	Indicates if the pivot bar is hidden while the cursor does not hover it. Include or exclude this flag to allow or disable the Auto-Hide feature. The Auto-Hide feature shows the pivot bar while the cursor hovers it, while the pivot bar is hidden, when cursor is outside of the pivot bar.
exPivotBarAllowValues	64	Specifies if the user can drop values of the columns to the pivot bar. By default, the control allow dragging columns as values, so actually, if the user drags the column to the second line, the value found on the column will compose the new column to be shown in the control's list. For instance, if you have a column named Region, it means that the list will display all regions, each region to be grouped in a single column. Here's a screen shot of value columns:  The Federal Shipping, Speedy Express and United Package columns indicates the values being found on the Shippers_CompanyName column.
exPivotBarAllowFormatAppearance	128	Specifies if the user can select a different appearance for columns or total/subtotal fields. Use the Add method of the FormatAppearances collection to add new predefined appearance to the context's list. By default, the control displays the list of appearances in the control's context menu as shown in the following screen shot:

Include or exclude this flag to allow or prevent applying the appearances using the control's context menu.

exPivotBarAllowFormatConditionalAp 1677

Specifies if the user can select conditional appearance for columns.

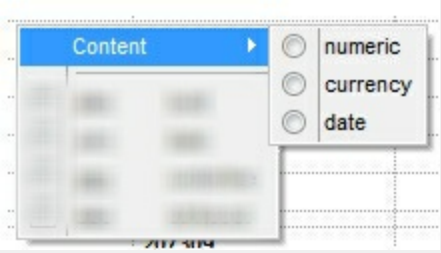
By default, the control displays the list of conditional appearances in the control's context menu as shown in the following screen shot:

Include or exclude this flag to allow or prevent applying the conditional appearances using the control's context menu.

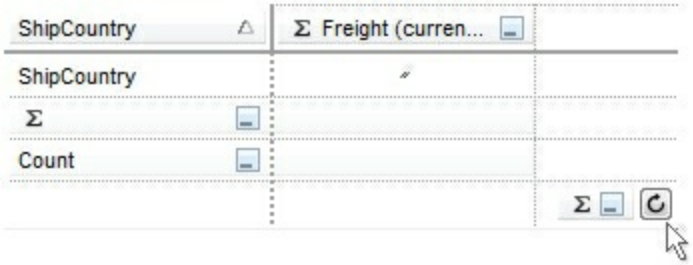
exPivotBarAllowFormatConten 256

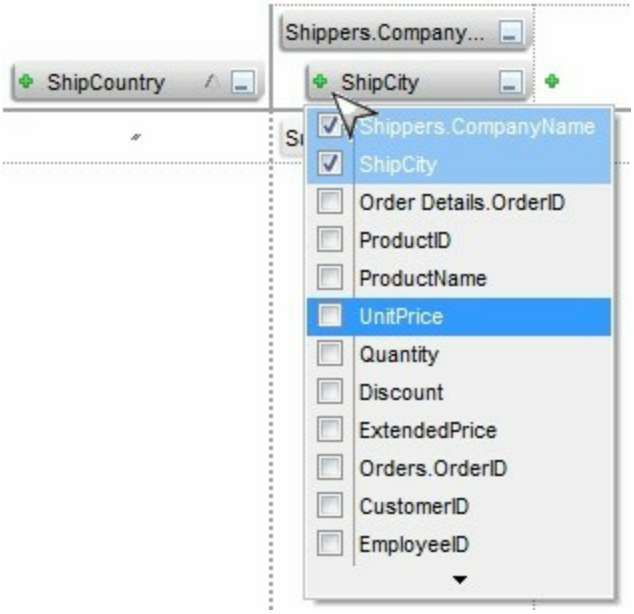
Specifies if the user can select the way the column's data is displayed. Use the Add method of the FormatContents collection to add new predefined functions to the context's list.

By default, the control displays the list of format functions in the control's context menu under the Content sub-menu as shown in the following screen shot:



Include or exclude this flag to allow or prevent applying the format functions using the control's context menu.

exPivotBarAutoUpdate	512	Indicates whether the control' list is automatically updated once the user drops objects in the pivot bar. By default, the control automatically updates-executes the control's layout as soon as the user drag and drop columns/aggregate functions to the control. In case this is a time-consuming depending on how large the data you provide, you can use this option to prevent executing the layout only when the user clicks the Refresh buttons as shown in the following screen shot:  Exclude this flag from the PivotBarVisible property, and so the Refresh button is shown in the bottom side of the control, and the changes will be applied to the control's list once the user clicks the Refresh button. Once the data is updated, the buttons shows as disabled until next change occurs. The control fires the LayoutStartChanging(exPivotDataLayoutExecute) / LayoutEndChanging(exPivotDataLayoutExecute) events once the control updates / executes the current layout.
		Indicates if the pivot bar allows Undo/Redo operations, if the user presses the CTRL+Z/Y. The control restores the previously layout once the user

exPivotBarAllowUndoRedo	1024	presses the CTRL + Z keys combination. The control restores the next layout once the user presses the CTRL + Y keys combination. Exclude this flag, and the Undo/Redo feature is disabled.
exPivotBarAllowResizeColumns	2048	Indicates if the user can resizes the columns in the pivot bar. This flag affects the columns to be displayed din the pivot bar, not the columns to be displayed in the control's list.
exPivotBarDefault	16781275	exPivotBarDefault. Indicates the default value of the PivotBarVisible property.
exPivotBarHideAddNew	4096	Prevents showing the add new buttons in the pivot bar. Clicking the Add New button displays a list of columns or aggregate functions that can be added in the current context. For instance, the Add New button in the Rows section, helps you to add more columns to Group-By rows, and a Add New button in the Columns section, adds new Group-By Columns or new Aggregate functions to be used. The following screen shot shows theAdd New button: 
exPivotBarContextSortAscending	65536	Shows the columns alphabetically in ascending order. This flag can be combined with exPivotBarContextSortReverse and so, the columns will be alphabetically displayed in descending order. The PivotColumnsSortOrder property specifies the sorting order of the columns to be shown on the Columns Floating Panel (

		PivotColumnsFloatBarVisible property)
exPivotBarContextSortReverse	1048576	Shows the columns in reverse order. This flag can be combined with exPivotBarContextSortAscending and so, the columns will be alphabetically displayed in descending order.
exPivotBarReadOnly	268435456	Makes the pivot bar read only, so no changes are allowed.

constants PivotColumnsSortOrderEnum

The PivotColumnsSortOrderEnum type specifies the way the columns are displayed. The PivotColumnsSortOrder property specifies the sorting order of the columns to be shown on the Columns Floating Panel (PivotColumnsFloatBarVisible property). The exPivotBarContextSortAscending, exPivotBarContextSortReverse flags of PivotBarVisible property specifies the sorting order of the columns to be shown on the pivot bar's context menu. The PivotColumnsSortOrderEnum type supports the following values:

Name	Value	Description
exPivotColumnsUnsorted	0	The columns are displayed the way they were loaded.
exPivotColumnsAscending	1	The columns are alphabetically displayed in ascending order.
exPivotColumnsDescending	2	The columns are alphabetically displayed in descending order.
exPivotColumnsReverse	3	The columns are displayed in reverse order.

constants ScrollBarEnum

The ScrollBarEnum type specifies the vertical or horizontal scroll bar in the control.

Name	Value	Description
exVScroll	0	Indicates the vertical scroll bar.
exHScroll	1	Indicates the horizontal scroll bar.
exHChartScroll	2	Reserved.

constants ScrollPartEnum

The ScrollPartEnum type defines the parts in the control's scrollbar.

Name	Value	Description
exLeftB1Part	32768	(L1) The first additional button, in the left or top area. By default, this button is hidden.
exLeftB2Part	16384	(L2) The second additional button, in the left or top area. By default, this button is hidden.
exLeftB3Part	8192	(L3) The third additional button, in the left or top area. By default, this button is hidden.
exLeftB4Part	4096	(L4) The forth additional button, in the left or top area. By default, this button is hidden.
exLeftB5Part	2048	(L5) The fifth additional button, in the left or top area. By default, this button is hidden.
exLeftBPart	1024	(<) The left or top button. By default, this button is visible.
exLowerBackPart	512	The area between the left/top button and the thumb. By default, this part is visible.
exThumbPart	256	The thumb part or the scroll box region. By default, the thumb is visible.
exUpperBackPart	128	The area between the thumb and the right/bottom button. By default, this part is visible.
exBackgroundPart	640	The union between the exLowerBackPart and the exUpperBackPart parts. By default, this part is visible.
exRightBPart	64	(>) The right or down button. By default, this button is visible.
exRightB1Part	32	(R1) The first additional button in the right or down side. By default, this button is hidden.
exRightB2Part	16	(R2) The second additional button in the right or down side. By default, this button is hidden.
exRightB3Part	8	(R3) The third additional button in the right or down side. By default, this button is hidden.
exRightB4Part	4	(R4) The forth additional button in the right or down side. By default, this button is hidden
		(R5) The fifth additional button in the right or down

exRightB5Part	2	side. By default, this button is hidden.
exRightB6Part	1	(R6) The sixth additional button in the right or down side. By default, this button is hidden.
exPartNone	0	No part.

constants ShowBranchRowsEnum

The ShowBranchRowsEnum type specifies way the rows can be displayed on the control's list. The ShowBranchRows property specifies the way the rows are arranged when the control shows the summarized data. Currently, the ShowBranchRowsEnum type supports the following values:

Name	Value	Description
exBranchTree	1	The exBranchTree mode displays data as a tree, and the +/- expanding buttons are shown. This flag cam be combined with the exBranchRowDivider or exBranchIncludeAggregate flag. The following screen shot shows the **exBranchTree** way: Σ Freight Shippers_CompanyName United Pac... ShipCountry ShipCity | Σ Freight | | Speedy Express | Federal Shipping Σ | **$207,155.00** | **$91,433.00** | **$52,284.00** | **$63,438.00** ⊟ Argentina Buenos Aires | $1,777.00 | $1,413.00 | $253.00 | $111.00 Σ | **$1,777.00** | **$1,413.00** | **$253.00** | **$111.00** ⊟ Austria Graz | $24,540.00 | $10,519.00 | $8,405.00 | $5,616.00 Salzburg | $3,190.00 | $1,581.00 | $660.00 | $949.00 Σ | **$27,730.00** | **$12,100.00** | **$9,065.00** | **$6,565.00** ⊟ Belgium Bruxelles | $1,087.00 | $453.00 | $558.00 | $76.00
exBranchCompact	2	The exBranchCompact mode displays data more compact, and the +/- expanding buttons are not shown. The exBranchCompact mode displays no subtotals (no hierarchy). The following screen shot shows the **exBranchCompact** way:

			Σ Freight Shippers_CompanyName United Pac...		
ShipCountry	ShipCity	Σ Freight		Speedy Express	Federal Shipping
Σ		**$207,155.00**	**$91,433.00**	**$52,284.00**	**$63,438.00**
Argentina	Buenos Ai...	$1,777.00	$1,413.00	$253.00	$111.00
Austria	Graz	$24,540.00	$10,519.00	$8,405.00	$5,616.00
	Salzburg	$3,190.00	$1,581.00	$660.00	$949.00
Belgium	Bruxelles	$1,087.00	$453.00	$558.00	$76.00
	Charleroi	$3,187.00	$1,957.00	$207.00	$1,023.00
Brazil	Campinas	$883.00	$124.00	$72.00	$687.00
	Resende	$547.00	$44.00	$359.00	$144.00
	Rio de Ja...	$4,313.00	$1,764.00	$2,166.00	$383.00
	São Paulo	$8,843.00	$5,604.00	$2,226.00	$1,013.00
Canada	Montréal	$4,124.00	$2,986.00	$465.00	$673.00

The exBranchCompact mode can not display the inside total/sub-totals fields.

exBranchRowDivider 16

The branch rows displays information on entire line. This flag may be combined with the exBranchTree flag.

The following screen shot shows the **exBranchTree + exBranchRowDivider** way:

			Σ Freight Shippers_CompanyName United Pac...		
ShipCountry	ShipCity	Σ Freight		Speedy Express	Federal Shipping
Σ		**$207,155.00**	**$91,433.00**	**$52,284.00**	**$63,438.00**
Argentina					
	Buenos Aires	$1,777.00	$1,413.00	$253.00	$111.00
Σ		**$1,777.00**	**$1,413.00**	**$253.00**	**$111.00**
Austria					
	Graz	$24,540.00	$10,519.00	$8,405.00	$5,616.00
	Salzburg	$3,190.00	$1,581.00	$660.00	$949.00
Σ		**$27,730.00**	**$12,100.00**	**$9,065.00**	**$6,565.00**
Belgium					
	Bruxelles	$1,087.00	$453.00	$558.00	$76.00
	Charleroi	$3,187.00	$1,957.00	$207.00	$1,023.00

The branch rows displays result of aggregate function. This flag may be combined with the exBranchTree flag.

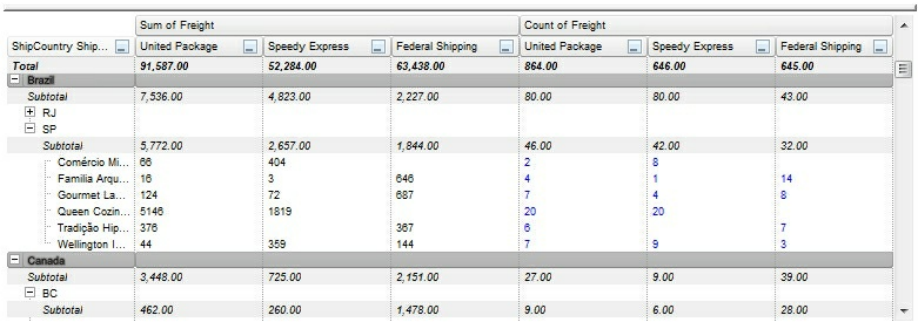
The following screen shot shows the **exBranchTree + exBranchIncludeAggregate** way:

exBranchIncludeAggregate 32

		Σ Freight		
ShipCountry ShipCity	Σ Freight	United Packa...	Speedy Expr...	Federal Shipping
Argentina	**$1,890.00**	**$438.00**	**$1,218.00**	**$234.00**
Buenos Aires	$1,890.00	$438.00	$1,218.00	$234.00
Austria	**$3,730.00**	**$3,086.00**	**$1,914.00**	**($1,270.00)**
Graz	$2,789.00	$2,452.00	$1,680.00	($1,343.00)
Salzburg	$941.00	$634.00	$234.00	$73.00
Belgium	**$3,098.00**	**$1,508.00**	**$516.00**	**$1,074.00**
Bruxelles	$1,085.00	$608.00	$219.00	$258.00
Charleroi	$2,013.00	$900.00	$297.00	$816.00
Brazil	**$3,316.00**	**$1,112.00**	**$382.00**	**$1,822.00**
Campinas	($1,098.00)	($2,638.00)	$849.00	$691.00
Resende	$1,441.00	$1,139.00	$174.00	$128.00
Rio de Janeiro	$2,931.00	$1,159.00	$1,251.00	$521.00
São Paulo	$42.00	$1,452.00	($1,892.00)	$482.00

constants ShowViewCompactEnum

The ShowViewCompactEnum type specifies the way the view compacts the displaying date. The ShowViewCompact property specifies whether the neighbor columns that generate the same values are compacted. The ShowViewCompactEnum type supports the following values:

Name	Value	Description
exViewNotCompact	0	The view does not compact displaying the data. By default, the ShowViewCompact property is exViewNotCompact, which means that the property has no effect. This flag can be combined with exViewCompactAggregates. 
exViewCompactSingleColumns	1	The view compacts the neighbor single pivot columns. This flag can be combined with exViewCompactKeepSettings or exViewCompactAggregates.
exViewCompactGroupByColumns	2	The view compacts the neighbor group-by pivot columns. This flag can be combined with exViewCompactKeepSettings or exViewCompactAggregates.
exViewCompact	3	The view compacts all columns. This flag can be combined with exViewCompactKeepSettings or exViewCompactAggregates. 

exViewCompactKeepSettings 16	Applies the same filter and sort settings to all neighbor pivot columns, while columns are shown as compacted (multiple aggregate functions in the same column). This flag has effect in OR combination with exViewCompactSingleColumns, exViewCompactGroupByColumns or exViewCompact.
exViewCompactAggregates 32	Compacts the aggregates functions, so the subtotals goes to the parent rows, rather than displaying them as child rows.

	Sum of Freight Count of Freight					
ShipCountry ShipRegion ShipName	United Package		Speedy Express		Federal Shipping	
Total	*91,587.00*	*864.00*	*52,284.00*	*646.00*	*63,438.00*	*645.00*
Brazil	***7,536.00***	***80.00***	***4,823.00***	***80.00***	***2,227.00***	***43.00***
RJ	*1,764.00*	*34.00*	*2,166.00*	*38.00*	*383.00*	*11.00*
SP	*5,772.00*	*46.00*	*2,657.00*	*42.00*	*1,844.00*	*32.00*
Comércio Mineiro	66	2	404	8		
Familia Arquibaldo	16	4	3	1	646	14
Gourmet Lanchonetes	124	7	72	4	687	8
Queen Cozinha	5146	20	1819	20		
Tradição Hipermercados	376	6			387	7
Wellington Importadora	44	7	359	9	144	3
Canada	***3,448.00***	***27.00***	***725.00***	***9.00***	***2,151.00***	***39.00***
BC	*462.00*	*9.00*	*260.00*	*6.00*	*1,478.00*	*28.00*
Bottom-Dollar Markets	462	9	260	6	1449	20
Laughing Bacchus Wine Cellars					29	8
Québec	*2,986.00*	*18.00*	*465.00*	*3.00*	*673.00*	*11.00*
Mère Paillarde	2986	18	465	3	673	11

constants SortOrderEnum

The SortOrderEnum type specifies how a column is sorted. The SortOrder property specifies the default sorting order for a pivot column. For instance, the SortOrder property specifies the default sort order when the user drags a column from the pivot columns floating bar to the control's pivot bar.

Name	Value	Description
SortNone	0	The column is not sorted.
SortAscending	1	The column is sorted ascending.
SortDescending	2	The column is sorted descending.

constants SortTypeEnum

The SortTypeEnum enumeration defines the types of sorting in the control. Use the SortType property to specifies the type of column's sorting.

Name	Value	Description
SortString	0	(Default) Values are sorted as strings.
SortNumeric	1	Values are sorted as numbers. Any non-numeric value is evaluated as 0.
SortDate	2	Values are sorted as dates. Group ranges are one day.
SortDateTime	3	Values are sorted as dates and times. Group ranges are one second.
SortTime	4	Values are sorted using the time part of a date and discarding the date. Group ranges are one second.
SortUserData	5	Reserved.
SortCellData	6	Reserved.
SortCellDataString	7	Reserved.

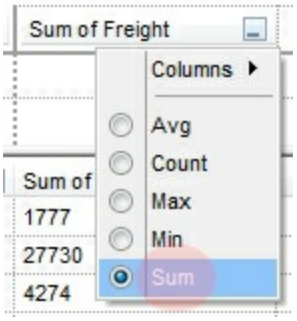
constants UIVisualThemeEnum

The UIVisualThemeEnum expression specifies the UI parts that the control can shown using the current visual theme. The UseVisualTheme property specifies whether the UI parts of the control are displayed using the current visual theme.

Name	Value	Description
exNoVisualTheme	0	exNoVisualTheme
exDefaultVisualTheme	16777215	exDefaultVisualTheme
exHeaderVisualTheme	1	exHeaderVisualTheme
exFilterBarVisualTheme	2	exFilterBarVisualTheme
exButtonsVisualTheme	4	exButtonsVisualTheme
exCalendarVisualTheme	8	exCalendarVisualTheme
exSliderVisualTheme	16	exSliderVisualTheme
exSpinVisualTheme	32	exSpinVisualTheme
exCheckBoxVisualTheme	64	exCheckBoxVisualTheme
exProgressVisualTheme	128	exProgressVisualTheme
exCalculatorVisualTheme	256	exCalculatorVisualTheme

Aggregate object

The Aggregate object defines an aggregate function to be used by the control. An aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning or measurement such as a set, a bag or a list. The Aggregates property gets access to Aggregate objects to define aggregate functions like **sum, avg, min, max, count**. The Aggregate objects are displayed (as radio buttons) on the control's context menu as shown:



The Aggregate object supports the following properties and methods:

Name	Description
Base	Specifies the name of the Base Aggregate function to be used by current Aggregate object.
Caption	Specifies the caption of the Aggregate to be displayed on the column.
FormatResult	Specifies the expression to be applied on the result of the aggregate function.
FormatValue	Specifies the expression to be applied on the value before passing to the aggregate function.
Key	Indicates the key of the Aggregate object.
Name	Specifies the name of the Aggregate to be displayed on the context menu.
ToolTip	Specifies the tooltip of the Aggregate to be displayed when the cursor hovers the object.

property Aggregate.Base as String

Specifies the name of the Base Aggregate function to be used by current Aggregate object.

Type	Description
String	A String expression that defines the base aggregate function to be used by the current Aggregate object. Any of the following values are supported. In case an unknown base function is used, the Aggregate object shows as disabled in the control's context menu.

The Base property indicates the base aggregate function to be used by the current Aggregate object. If the Base property points to an unknown value, the Aggregate object shows as disabled in the control's context menu.

The Base property could be one on the following:

- **sum**, summation is the operation of adding a sequence of numbers; the result is their sum or total
- **min**, minimum is the smallest value
- **max**, maximum is the largest value
- **count**, counts the number of objects in the set
- **avg**, average is the arithmetic mean, which means sum of all numbers divided by the count.

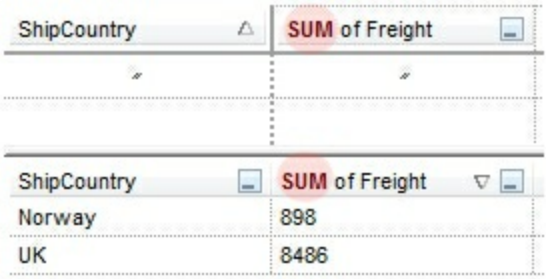
property Aggregate.Caption as String

Specifies the caption of the Aggregate to be displayed on the column.

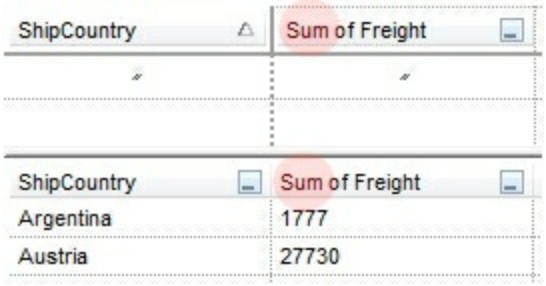
Type	Description
String	A String expression that defines the HTML Caption to be displayed on the Column, if the Aggregate function is applied to the column.

The Caption property indicates the HTML caption to be displayed on the column's header when the current Aggregate object is applied to the column. In other words, the Caption property is displayed in the column's header when the column is drag and drop to the control's pivot bar. The FormatPivotHeader property defines the format of the caption to be displayed on the column's header. The **caggregate** keyword in the FormatPivotHeader property defines the Aggregate's Caption property. The Name property of the Aggregate object defines the name to be displayed on the control's context menu. You can use the <img> HTML tag to display icons or pictures. Use the Images or HTMLPicture method to add icons or pictures to control.

The following screen shot shows the new Caption (SUM):



The following screen shot shows the default Caption (Sum):



The Caption property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a

piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.

- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

  gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

  outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

  shadow

  or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

property Aggregate.FormatResult as String

Specifies the expression to be applied on the result of the aggregate function.

Type	Description
String	A String expression that defines the formula to convert/format the result. If missing or empty, the result suffers no changes.

By default, the FormatResult property is empty. The FormatResult property can be used to convert or format the result of the aggregate functions. The FormatValue property may be used if you need to convert the value instead the result, else you can use the FormatResult property to convert/format the result directly.

For instance:

- you want to display the result multiplied by 1.19 to show the VAT, so all you need is to add a new aggregate function like *Aggregates.Add("vat", "sum").FormatResult = "value * 1.19"*, so the result is multiplied by 1.19.
- *Aggregates.Add("colors", "sum").FormatResult = "(value < 500 ? '<fgcolor=FF0000>' : '') + value"*, shows the values under 500 in red.

The **value** keyword in the FormatResult property indicates the value/result to be converted.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- ***/*** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

  "expression switch (default,c1,c2,c3,...,cn)"

  , where the c1, c2, ... are constant elements, and the default is a constant element

being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

  "expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

  If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
  - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8

specifies the cells that contains string values.

Here's few predefined types:

- 0 - empty (not initialized)
- 1 - null
- 2 - short
- 3 - long
- 4 - float
- 5 - double
- 6 - currency
- 7 - date
- 8 - string
- 9 - object
- 10 - error
- 11 - boolean
- 12 - variant
- 13 - any
- 14 - decimal
- 16 - char
- 17 - byte
- 18 - unsigned short
- 19 - unsigned long
- 20 - long on 64 bits
- 21 - unsigned long on 64 bites

- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format " displays

1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as

appropriate for proper names

- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

property Aggregate.FormatValue as String

Specifies the expression to be applied on the value before passing to the aggregate function.

Type	Description
String	A String expression that defines the formula to be applied on the input value, so it can be passed to the aggregate function. If missing, the value itself is passed to the Aggregate function.

The FormatValue property defines the expression to be applied on individual values before passing to the Aggregate function. The Base property indicates the base aggregate function, in most cases you will use the "sum" function as explained bellow. The FormatValue property may be used if you need to convert the value instead the result, else you can use the FormatResult property to convert/format the result directly.

For instance:

- you want to get the total for negative values or to count the positive value only. In this case, you can add a new Aggregate object such as Aggregates.Add("negative", "sum").FormatValue = "value < 0 ? value : 0", and so the negative Aggregate function gets the total of negative values only.
- The Aggregates.Add("positive", "sum").FormatValue = "value < 0 ? 0 : 1", counts the number of positive values.

The **value** keyword in the FormatValue property indicates the value to be converted.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- **/** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2

- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
  - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

  - 0 - empty (not initialized)
  - 1 - null
  - 2 - short
  - 3 - long
  - 4 - float
  - 5 - double
  - 6 - currency
  - 7 - date
  - 8 - string
  - 9 - object
  - 10 - error
  - 11 - boolean
  - 12 - variant
  - 13 - any
  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long
  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2

- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
  - *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
  - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
    - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
    - 1 - Negative sign, number; for example, -1.1
    - 2 - Negative sign, space, number; for example, - 1.1
    - 3 - Number, negative sign; for example, 1.1-
    - 4 - Number, space, negative sign; for example, 1.1 -
  - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string

- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)

- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

property Aggregate.Key as String

Indicates the key of the Aggregate object.

Type	Description
String	A String expression that specifies the unique key of the Aggregate object. The Key must contain alpha numeric characters. Any other character is not included in the key.

The Aggregates object holds a collection of Aggregate objects, each object must have an unique key that identifies the Aggregate object. The Name property of the Aggregate object defines the name to be displayed on the control's context menu. The Caption property indicates the HTML caption to be displayed on the column's header when the current Aggregate object is applied to the column. The Key parameter of the Add method can be used to assign a key at adding time.

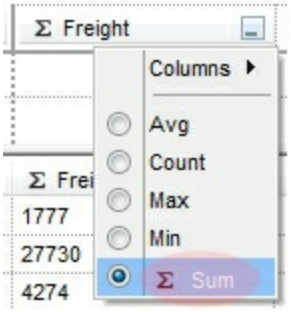
property Aggregate.Name as String

Specifies the name of the Aggregate to be displayed on the context menu.

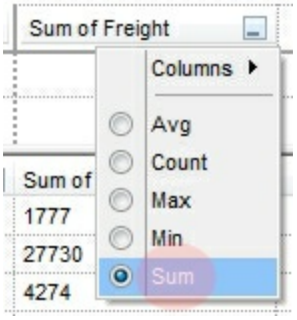
Type	Description
String	A String expression that defines the HTML name to be displayed on the control's context menu.

The Name property defines the HTML caption to be displayed on the control's context menu. The Caption property indicates the HTML caption to be displayed on the column's header when the current Aggregate object is applied to the column. You can use the <img> HTML tag to display icons or pictures. Use the Images or HTMLPicture method to add icons or pictures to control.

The following screen show shows the new Name (<img>1</img> Sum):



The following screen show shows the default Name (Sum):



The Caption property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor

element. The *FormatAnchor* property customizes the visual effect for anchor elements.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display

bold in HTML caption you can use bold

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

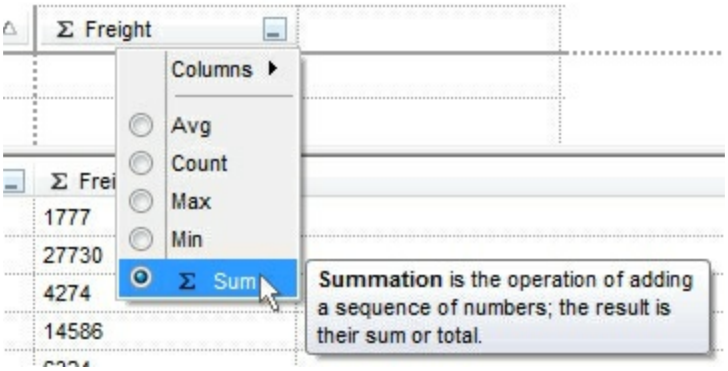
outline anti-aliasing

property Aggregate.ToolTip as String

Specifies the tooltip of the Aggregate to be displayed when the cursor hovers the object.

Type	Description
String	A String expression that defines the HTML tooltip to be shown when the cursor hovers the Aggregate object in the control's context menu.

By default, the ToolTip property is empty. Use the ToolTip property to assign a tooltip to each Aggregate object. The ToolTip property supports the HTML format as listed bellow. The Name property of the Aggregate object defines the name to be displayed on the control's context menu. You can use the <img> HTML tag to display icons or pictures. Use the Images or HTMLPicture method to add icons or pictures to control. You can use the ShowToolTip method to display a custom tooltip.



The ToolTip property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with

a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the

text such as: Text with subscript

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

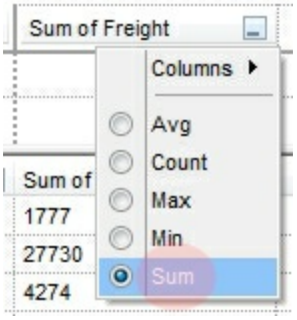
shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

Aggregates object

The Aggregates collection holds the aggregate functions the user can use and display in the pivot control. The Aggregates property gets access to Aggregate objects to define aggregate functions like **sum, avg, min, max, count**. The Aggregate objects are displayed (as radio buttons) on the control's context menu as shown:



The Aggregates collection supports the following properties and methods:

Name	Description
Add	Adds an Aggregate function and returns a reference to the newly created object.
Clear	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
Item	Returns a specific Aggregate function giving its key.
Remove	Removes a specific member from the collection.

method Aggregates.Add (Key as String, Base as Variant, [Name as Variant], [Caption as Variant])

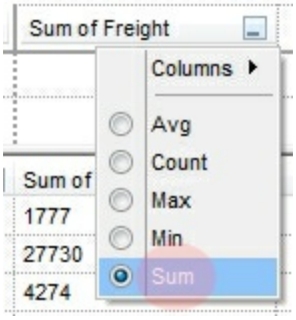
Adds an Aggregate function and returns a reference to the newly created object.

Type	Description
Key as String	A String expression that specifies the unique key to identify the Aggregate object. The Key parameter should include only alpha-numeric characters. Any other characters are not included. The Key property specifies the Aggregate's key.
Base as Variant	A String expression that defines the base aggregate function to be used by the Aggregate object. The valid values are: "sum", "min", "max", "count" or "avg". Any other value makes the Aggregate object to be shown as disabled when showing in the control's context menu.
Name as Variant	A String expression that indicates the HTML caption to be shown on the control's context menu
Caption as Variant	A String expression that indicates the HTML caption to be shown on the column's header.

Return	Description
Aggregate	An Aggregate object being created, that holds the newly object.

The Add method adds an Aggregate object to the Aggregates collection. An aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning or measurement such as a set, a bag or a list. The Aggregate object is identified by an unique key. By default, the Aggregates collection contains the "sum", "min", "max", "count" or "avg" Aggregate objects. Use the FormatValue/FormatResult property to convert/format the value/result of the Aggregate function. The PivotColumns or PivotTotals property to display aggregate functions on the columns.

The Aggregate objects are shown on the control's context menu as shown bellow:



The Base parameter could be one on the following:

- **sum**, summation is the operation of adding a sequence of numbers; the result is their sum or total
- **min**, minimum is the smallest value
- **max**, maximum is the largest value
- **count**, counts the number of objects in the set
- **avg**, average is the arithmetic mean, which means sum of all numbers divided by the count.

method Aggregates.Clear ()

Removes all objects in a collection.

Type	Description

The Clear method removes all elements in the Aggregates collection. Use the Remove method to remove an individual Aggregate object from the Aggregates collection. The Item property of the Aggregates collection accesses an Aggregate object giving its key. The Count property of the Aggregates collection counts the number of Aggregate objects in the Aggregates collection.

property Aggregates.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A Long expression that specifies the number of Aggregate objects.

The Count property counts the number of objects in the Aggregates collection. The Item property accesses an Aggregate object giving its key. You can use the **for each** statement to enumerate all Aggregate objects in the Aggregates collection. The Clear method removes all objects in the collection.

property Aggregates.Item (Key as Variant) as Aggregate

Returns a specific Aggregate function giving its key.

Type	Description
Key as Variant	A String expression that specifies the key of the Aggregate object to be retrieved
Aggregate	An Aggregate object to be accessed.

The Item property accesses an Aggregate object giving its key. The Count property counts the number of objects in the Aggregates collection. You can use the **for each** statement to enumerate all Aggregate objects in the Aggregates collection. For instance, the Aggregates.Item("sum").FormatResult = "currency(value)", makes a sum column to display its content as a currency, by default.

method Aggregates.Remove (Key as Variant)

Removes a specific member from the collection.

Type	Description
Key as Variant	A String expression that specifies the key of the Aggregate object to be removed. No error is returned if no Aggregate with specified key is found.

Use the Remove method to remove an individual Aggregate object from the Aggregates collection. The Clear method removes all elements in the Aggregates collection. The Item property of the Aggregates collection accesses an Aggregate object giving its key. The Count property of the Aggregates collection counts the number of Aggregate objects in the Aggregates collection.

Appearance object

The component lets the user changes its visual appearance using **skins**, each one providing an additional visual experience that enhances viewing pleasure. Skins are relatively easy to build and put on any part of the control. The Appearance object holds a collection of skins. The Appearance object supports the following properties and methods:

Name	Description
Add	Adds or replaces a skin object to the control.
Clear	Removes all skins in the control.
Remove	Removes a specific skin from the control.
RenderType	Specifies the way colored EBN objects are displayed on the component.

method Appearance.Add (ID as Long, Skin as Variant)

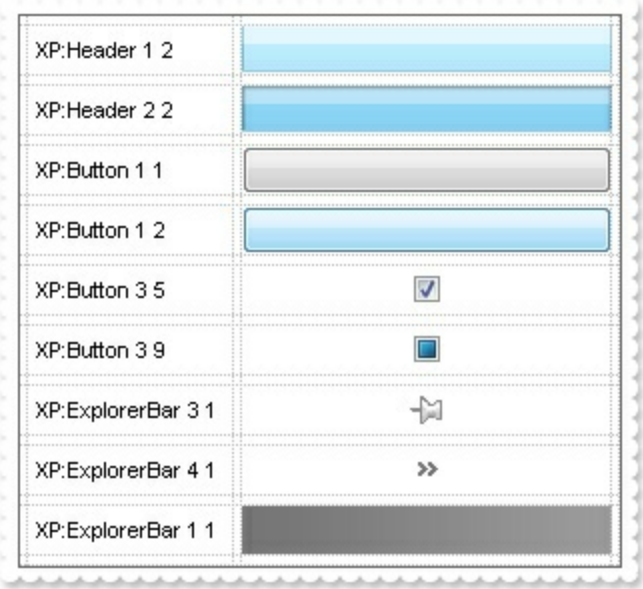
Adds or replaces a skin object to the control.

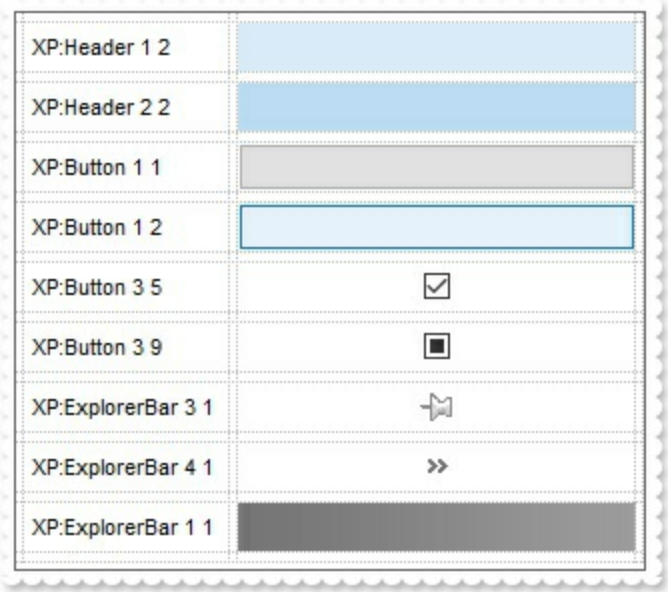
Type	Description
ID as Long	A Long expression that indicates the index of the skin being added or replaced. The value must be between 1 and 126, so Appearance collection should holds no more than 126 elements.
	The Skin parameter of the Add method can a STRING as explained bellow, a BYTE[] / safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. You can use the BYTE[] / safe arrays of VT_I1 or VT_UI1 option when using the EBN file directly in the resources of the project. For instance, the VB6 provides the LoadResData to get the safe array o bytes for specified resource, while in VB/NET or C# the internal class Resources provides definitions for all files being inserted. (ResourceManager.GetObject("ebn", resourceCulture)) If the Skin parameter points to a string expression, it can be one of the following: • A path to the skin file (*.EBN). The ExButton component or ExEBN tool can be used to create, view or edit EBN files. For instance, "C:\Program Files\Exontrol\ExButton\Sample\EBN\MSOffice-Ribbon\msor_frameh.ebn" • A BASE64 encoded string that holds the skin file (*.EBN). Use the ExImages tool to build BASE 64 encoded strings of the skin file (*.EBN). The BASE64 encoded string starts with "gBFLBCJw..." • An Windows XP theme part, if the Skin parameter starts with "**XP:**". Use this option, to display any UI element of the Current Windows XP Theme, on any part of the control. In this case, the syntax of the Skin parameter is: "XP:ClassName Part State" where the ClassName defines the window/control class name in the Windows XP Theme, the Part indicates a long expression that defines the part, and the State indicates the state of the part to be shown. All known values for window/class, part and start are defined at

the end of this document. For instance the "XP:Header 1 2" indicates the part 1 of the Header class in the state 2, in the current Windows XP theme.

The following screen shots show a few Windows XP Theme Elements, running on Windows Vista and Windows 10:

Skin as Variant

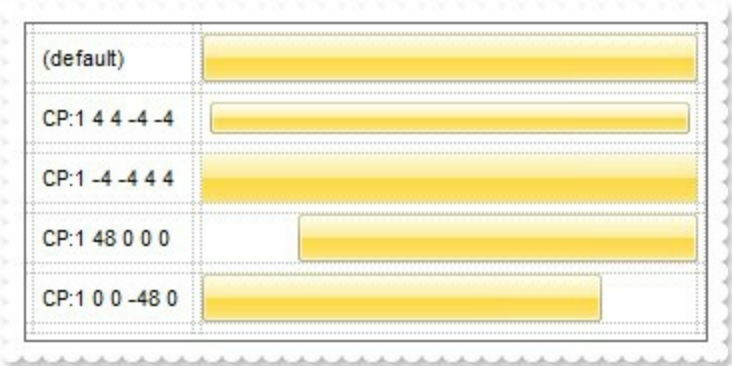




- A copy of another skin with different coordinates (position, size), if the Skin parameter starts with "**CP**:". Use this option, to display the EBN, using different coordinates (position, size). By default, the EBN skin object is rendered on the part's client area. Using this option, you can display the same EBN, on a different position / size. In this case, the syntax of the Skin parameter is: "CP:ID Left Top Right Bottom"

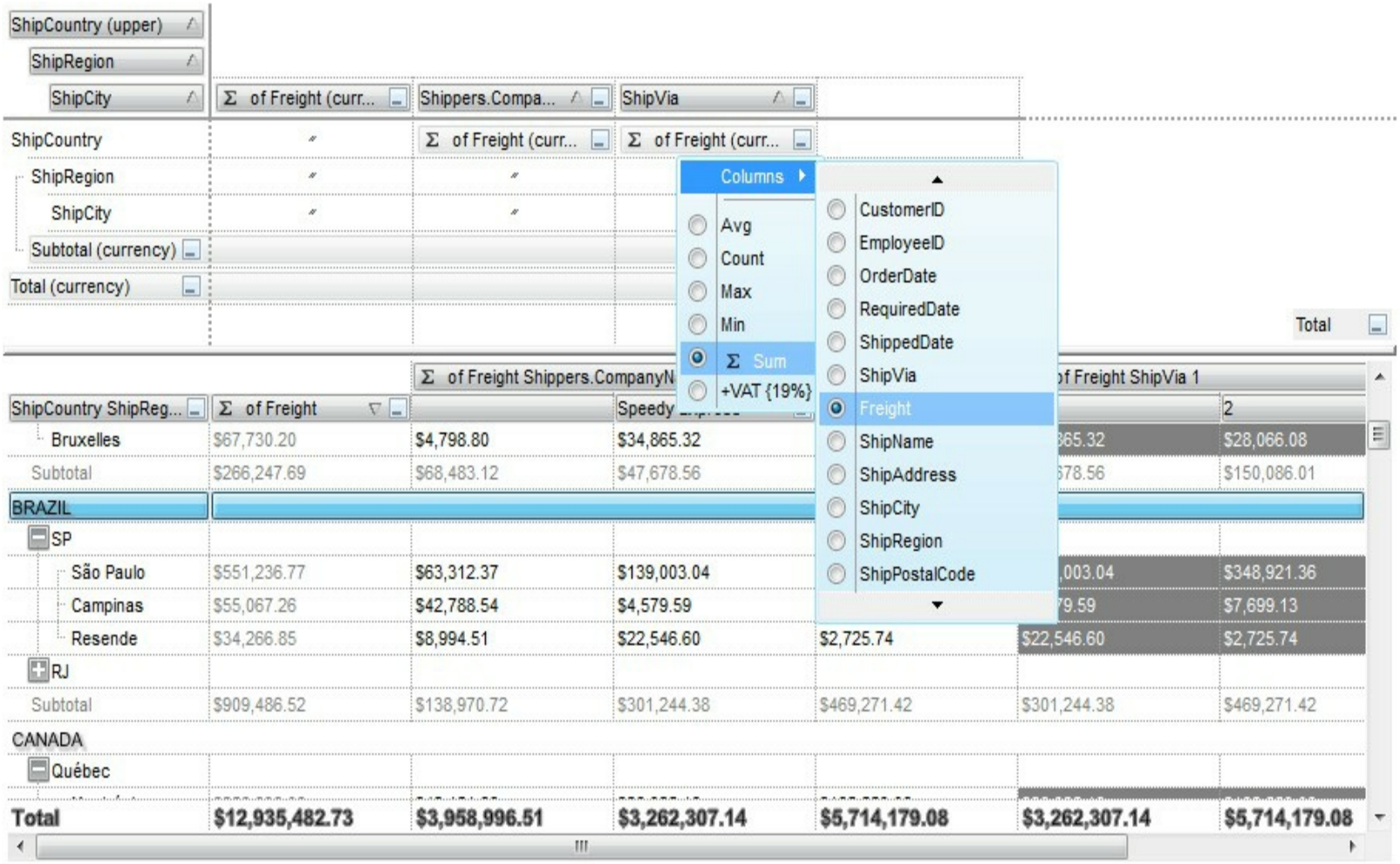
where the ID is the identifier of the EBN to be used (it is a number that specifies the ID parameter of the Add method), Left, Top, Right and Bottom parameters/numbers specifies the relative position to the part's client area, where the EBN should be rendered. The Left, Top, Right and Bottom parameters are numbers (negative, zero or positive values, with no decimal), that can be followed by the D character which indicates the value according to the current DPI settings. For instance, "CP:1 -2 -2 2 2", uses the EBN with the identifier 1, and displays it on a 2-pixels wider rectangle no matter of the DPI settings, while "CP:1 -2D -2D 2D 2D" displays it on a 2-pixels wider rectangle if DPI settings is 100%, and on on a 3-pixels wider rectangle if DPI settings is 150%.

The following screen shot shows the same EBN being displayed, using different CP: options:



Return	Description
Boolean	A Boolean expression that indicates whether the new skin was added or replaced.

Use the Add method to add or replace skins to the control. Use the VisualDesign property to define the control's visual appearance at design mode. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part. Use the Remove method to remove a specific skin from the control. Use the Clear method to remove all skins in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while init the control. Use the Refresh method to refresh the control.



The identifier you choose for the skin is very important to be used in the background properties like explained bellow. Shortly, the color properties uses 4 bytes (DWORD, double WORD, and so on) to hold a RGB value. More than that, the first byte (most significant byte in the color) is used only to specify system color. if the first bit in the byte is 1, the rest of bits indicates the index of the system color being used. So, we use the last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. So, since the 7 bits can cover 127 values, excluding 0, we have 126 possibilities to store an identifier in that byte. This way, a DWORD expression indicates the background color stored in RRGGBB format and the index of the skin (ID parameter) in the last 7 bits in the high significant byte of the color. For instance, the BackColor = BackColor Or &H2000000 indicates that we apply the skin with the index 2 using the old color, to the object that BackColor is applied.

The skin method may change the visual appearance for the following parts in the control:

- control's border, Appearance property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- **selected item** or cell, SelBackColor property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltip, and so on, Background property

For instance, the following VB sample changes the visual appearance for the selected item. The SelBackColor property indicates the selection background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the SelBackColor property to indicates the index of the skin that we want to use. The sample applies the "" to the selected item(s):

```
With Pivot1
   With .VisualAppearance
      .Add &H23, App.Path + "\selected.ebn"
   End With
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = &H23000000
End With
```

The sample adds the skin with the index 35 (Hexa 23), and applies to the selected item using the SelBackColor property.

The following C++ sample applies a new appearance to the selected item(s):

```
#include "Appearance.h"
m_pivot.GetVisualAppearance().Add( 0x23,
COleVariant(_T("D:\\Temp\\ExPivot_Help\\selected.ebn")) );
m_pivot.SetSelBackColor( 0x23000000 );
m_pivot.SetSelForeColor( 0 );
```

The following VB.NET sample applies a new appearance to the selected item(s):

```
With AxPivot1
   With .VisualAppearance
      .Add(&H23, "D:\Temp\ExPivot_Help\selected.ebn")
   End With
   .SelForeColor = Color.Black
   .Template = "SelBackColor = 587202560"
End With
```

The VB.NET sample uses the Template property to assign a new value to the SelBackColor property. The 587202560 value represents &23000000 in hexadecimal.

The following C# sample applies a new appearance to the selected item(s):

```
axPivot1.VisualAppearance.Add(0x23, "D:\\Temp\\ExPivot_Help\\selected.ebn");
```

```
axPivot1.Template = "SelBackColor = 587202560";
```

The following VFP sample applies a new appearance to the selected item(s):

```
With thisform.Pivot1
   With .VisualAppearance
      .Add(35, "D:\Temp\ExPivot_Help\selected.ebn")
   EndWith
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = .587202560
EndWith
```

The 587202560 value represents &23000000 in hexadecimal. The 32 value represents &23 in hexadecimal

The screen shot was generated using the following template:

```
BeginUpdate

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl

Images("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0plUrlkt

Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl



Font
{
   Name = "Tahoma"
}

VisualAppearance
{
   ' Header

Add(1,"gBFLBCJwBAEHhEJAEGg4BawDg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAGEYxR
```

```
    ' HeaderFilterBarButton

Add(2,"gBFLBCJwBAEHhEJAEGg4BAQEg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAGEYxR

Add(3,"gBFLBCJwBAEHhEJAEGg4BBAEg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAGEYxR

    ' SelectedItem
    Add(4,
"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK

    ' Marks a cell

Add(5,"gBFLBCJwBAEHhEJAEGg4BF4Gg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAga

Add(6,"gBFLBCJwBAEHhEJAEGg4BaAFg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA

}
BackColorHeader = 16777216          '0x01BBGGRR
BackColorSortBarCaption = 33488896     '0x01BBGGRR
FilterBarBackColor = 16777216        '0x01BBGGRR
Background(0) = 33554432             '0x02BBGGRR
Background(1) = 50331648             '0x03BBGGRR
Background(2) = 67108864             '0x04BBGGRR
Background(3) = 100663296            '0x06BBGGRR
Background(8) = 67108864             '0x04BBGGRR
Background(9) = 67108864             '0x04BBGGRR
Background(10) = 100663296              '0x06BBGGRR
Background(11) = 100663296              '0x06BBGGRR
Background(12) = 100663296              '0x06BBGGRR
Background(13) = 100663296              '0x06BBGGRR
Background(14) = 100663296              '0x06BBGGRR
Background(15) = 16777216            '0x01BBGGRR
SelBackColor = 67108864              '0x04BBGGRR
BackColorSortBar = RGB(61,101,183)
FilterBarForeColor = RGB(255,255,255)
```

```
ForeColorHeader = RGB(255,255,255)
ForeColorSortBar = RGB(255,255,255)
SelForeColor = 0

SortBarVisible = True
MarkSearchColumn = False
LinesAtRoot = 1
ForeColor = RGB(0,0,255)
BackColor = RGB(255,255,255)
BackColorLevelHeader = RGB(255,255,255)
DrawGridLines = -1
ScrollBySingleLine = True
HasLines = 2
HasButtons = 3
CheckImage(1) = 4
CheckImage(0) = 5
CheckImage(2) = 6
Chart
{
   DrawGridLines = -1
   BackColor = RGB(255,255,255)
   BackColorLevelHeader = 16777216      '0x01BBGGRR
   ForeColorLevelHeader = RGB(255,255,255)
   ScrollBar = False
   Bars
   {
      AddShapeCorner(1234,1)
      AddShapeCorner(1235,2)
      Add("Custom")
      {
         Color = RGB(255,0,0)
         Shape = 19
         Pattern = 2
         StartShape = 1234
         StartColor = RGB(255,0,0)
         EndShape = 1235
         EndColor = RGB(255,0,0)
```

```
      }
   }
}
Columns
{
   "Task"
   {
      HeaderBold = True
      DisplayFilterButton = True
      DisplayFilterDate = True
      Width = 196
   }
   1
   {
      AllowSizing = False
      HTMLCaption = "1 First"
      Def(0) = True
      LevelKey = 1
      Width = 25
      Alignment = 1
   }
   2
   {
      AllowSizing = False
      HTMLCaption = "2 Second"
      Def(0) = True
      LevelKey = 1
      Width = 25
      Alignment = 1
   }
   3
   {
      AllowSizing = False
      HTMLCaption = "3 Third"
      Def(0) = True
      LevelKey = 1
      Width = 25
```

```
      PartialCheck = True
      Alignment = 1
   }
   """
   {
      LevelKey = 1
      Width = 20
   }
   """
   {
      Position = 0
      Def(2) = True
      Width = 16
   }

}
Chart
{
   FirstVisibleDate = "5/29/2005"
}
Items
{
   Dim h, h1,hx
   h = AddItem(" exPivot Add an advanced pivot chart to your application.")
   CellTooltip(h,0) = "You can have a HTML multiple lines tooltip for any cell in the tree."
   CellCaptionFormat(h,0) = 1
   CellSingleLine(h,0) = False
   CellImages(h,1) = "1,2,3,"
   CellHAlignment(h,1) = 2
   CellMerge(h,0) = 1
   CellMerge(h,0) = 2
   CellMerge(h,0) = 3
   AddBar(h,"Progress","5/30/2005","6/4/2005",1,"
- TODO: project -")
   AddBar(h,"Deadline","5/29/2005 16:00","6/2/2005",2)
   AddBar(h,"Deadline","6/4/2005 07:00","6/10/2005",3)
```

```
h1 = InsertItem(h,,"Project Sumarry1")
CellHasCheckBox(h1,0) = True
CellImage(h1,0) = 1
AddBar(h1,"Project Summary","5/31/2005","6/15/2005"")
AddBar(h1,"Milestone","5/30/2005","5/31/2005","M")
AddBar(h1,"Milestone","6/16/2005","6/17/2005","E")

h1 = InsertItem(h,,"Task...Split")
CellHasCheckBox(h1,0) = True
CellState(h1,0) = 1
CellImage(h1,0) = 2
AddBar(h1,"Task","6/1/2005","6/4/2005","S")
AddBar(h1,"Split","6/4/2005","6/6/2005","Split")
AddBar(h1,"Task","6/6/2005","6/12/2005","E")

ExpandItem(h) = True

h = AddItem("")
CellCaption(h,1) = "Custom icons ..."
CellCaptionFormat(h,1) = 1
ItemDivider(h) = 1
ItemHeight(h) = 28
ItemDividerLine(h) = 3
CellHAlignment(h,1) = 1
SelectableItem(h) = False
CellPicture(h,1) =
"gBHJJGHA5MIqAAXAD3AENhozhpmhqZhrMhr/h0QGcQM0QTMQZkQf8QAESGcSM0STM

AddBar(h,"Custom","5/31/2005","6/4/2005")
ItemBackColor(h) = 100663296

h = AddItem("Root 2")
CellImages(h,0) = "2,3"
ItemBold(h) = True
CellMerge(h,0) = 1
CellMerge(h,0) = 2
CellMerge(h,0) = 3
```

```
    h1 = InsertItem(h,,"Task 1")
    AddBar(h1,"Task","6/4/2005","6/5/2005","S")
    AddBar(h1,"Split","6/5/2005","6/8/2005","Split")
    AddBar(h1,"Task","6/8/2005","6/10/2005","E")
    AddBar(h1,"","5/30/2005 12:00","6/3/2005 11:00",,"some text")
    ItemBar(h1,,7) = 83886080
'   ItemBar(h1,,8) = RGB(255,255,255)
    ItemBar(h1,,6) = "This is a bit of text that should occur when the cursor hovers the bar
or the text."

    h1 = InsertItem(h,,"Task 2")

    ExpandItem(h) = true

}

EndUpdate
```

Starting with **Windows XP**, the following table shows how the common controls are broken into parts and states:

Control/ClassName	Part	States
BUTTON	BP_CHECKBOX = 3	CBS_UNCHECKED 1 CBS_UNCHECKE CBS_UNCHECKED = 3 CBS_UNCHECKED = 4 CBS_CHECKED 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRESS CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
		PBS_NORMAL = 1 = 2 PBS_PRESSED

	BP_PUSHBUTTON = 1	PBS_DISABLED = PBS_DEFAULTED =
	BP_RADIOBUTTON = 2	RBS_UNCHECKED 1 RBS_UNCHECKE RBS_UNCHECKED = 3 RBS_UNCHECKED = 4 RBS_CHECKED 5 RBS_CHECKEDH RBS_CHECKEDPR RBS_CHECKEDDIS
	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1
COMBOBOX	CP_DROPDOWNBUTTON = 1	CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED =
EDIT	EP_CARET = 2	
	EP_EDITTEXT = 1	ETS_NORMAL = 1 2 ETS_SELECTED ETS_DISABLED = ETS_FOCUSED = 5 ETS_READONLY = ETS_ASSIST = 7
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
	EBP_HEADERCLOSE = 2	EBHC_NORMAL = EBHC_HOT = 2 EBHC_PRESSED =
	EBP_HEADERPIN = 3	EBHP_NORMAL = EBHP_HOT = 2 EBHP_PRESSED = EBHP_SELECTEDN 4 EBHP_SELECTE EBHP_SELECTEDP 6
	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSE
	EBP_NORMALGROUPBACKGROUND = 5	
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_NORMAL = EBNGC_HOT = 2 EBNGC_PRESSED

	EBP_NORMALGROUPEXPAND = 7	EBNGE_NORMAL = EBNGE_HOT = 2 EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
	EBP_SPECIALGROUPCOLLAPSE = 10	EBSGC_NORMAL = EBSGC_HOT = 2 EBSGC_PRESSED
	EBP_SPECIALGROUPEXPAND = 11	EBSGE_NORMAL = EBSGE_HOT = 2 EBSGE_PRESSED
	EBP_SPECIALGROUPHEAD = 12	
HEADER	HP_HEADERITEM = 1	HIS_NORMAL = 1 H 2 HIS_PRESSED =
	HP_HEADERITEMLEFT = 2	HILS_NORMAL = 1 = 2 HILS_PRESSEI
	HP_HEADERITEMRIGHT = 3	HIRS_NORMAL = 1 = 2 HIRS_PRESSE
	HP_HEADERSORTARROW = 4	HSAS_SORTEDUP HSAS_SORTEDDC
LISTVIEW	LVP_EMPTYTEXT = 5	
	LVP_LISTDETAIL = 3	
	LVP_LISTGROUP = 2	
	LVP_LISTITEM = 1	LIS_NORMAL = 1 L 2 LIS_SELECTED = LIS_DISABLED = 4 LIS_SELECTEDNO 5
	LVP_LISTSORTEDDETAIL = 4	
MENU	MP_MENUBARDROPDOWN = 4	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
	MP_MENUBARITEM = 3	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
	MP_CHEVRON = 5	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
		MS_NORMAL = 1 MS_SELECTED = 2

	MP_MENUDROPDOWN = 2	MS_DEMOTED = 3
	MP_MENUITEM = 1	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
	MP_SEPARATOR = 6	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
MENUBAND	MDP_NEWAPPBUTTON = 1	MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE
	MDP_SEPERATOR = 2	
PAGE	PGRP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	PGRP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	PGRP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	PGRP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
PROGRESS	PP_BAR = 1	
	PP_BARVERT = 2	
	PP_CHUNK = 3	
	PP_CHUNKVERT = 4	
REBAR	RP_BAND = 3	
	RP_CHEVRON = 4	CHEVS_NORMAL = CHEVS_HOT = 2 CHEVS_PRESSED
	RP_CHEVRONVERT = 5	
	RP_GRIPPER = 1	
	RP_GRIPPERVERT = 2	
		ABS_DOWNDISAB

SCROLLBAR	SBP_ARROWBTN = 1	ABS_DOWNHOT, ABS_DOWNNORM ABS_DOWNPRESS ABS_UPDISABLED ABS_UPHOT, ABS_UPNORMAL, ABS_UPPRESSED, ABS_LEFTDISABLI ABS_LEFTHOT, ABS_LEFTNORMA ABS_LEFTPRESSE ABS_RIGHTDISAB ABS_RIGHTHOT, ABS_RIGHTNORM ABS_RIGHTPRESS
	SBP_GRIPPERHORZ = 8 SBP_GRIPPERVERT = 9	
	SBP_LOWERTRACKHORZ = 4	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_LOWERTRACKVERT = 6	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNHORZ = 2	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNVERT = 3	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKHORZ = 5	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKVERT = 7	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
		SZB_RIGHTALIGN

	SBP_SIZEBOX = 10	SZB_LEFTALIGN =
SPIN	SPNP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	SPNP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	SPNP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	SPNP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
STARTPANEL	SPP_LOGOFF = 8	
	SPP_LOGOFFBUTTONS = 9	SPLS_NORMAL = SPLS_HOT = 2 SPLS_PRESSED =
	SPP_MOREPROGRAMS = 2	
	SPP_MOREPROGRAMSARROW = 3	SPS_NORMAL = 1 = 2 SPS_PRESSED
	SPP_PLACESLIST = 6	
	SPP_PLACESLISTSEPARATOR = 7	
	SPP_PREVIEW = 11	
	SPP_PROGLIST = 4	
	SPP_PROGLISTSEPARATOR = 5	
	SPP_USERPANE = 1	
	SPP_USERPICTURE = 10	
STATUS	SP_GRIPPER = 3	
	SP_PANE = 1	
	SP_GRIPPERPANE = 2	
TAB	TABP_BODY = 10	
	TABP_PANE = 9	
	TABP_TABITEM = 1	TIS_NORMAL = 1 T 2 TIS_SELECTED = TIS_DISABLED = 4 TIS_FOCUSED = 5

		TIBES_NORMAL =
		TIBES_HOT = 2
	TABP_TABITEMBOTHEDGE = 4	TIBES_SELECTED
		TIBES_DISABLED
		TIBES_FOCUSED =
		TILES_NORMAL =
		TILES_HOT = 2
	TABP_TABITEMLEFTEDGE = 2	TILES_SELECTED
		TILES_DISABLED =
		TILES_FOCUSED =
		TIRES_NORMAL =
		TIRES_HOT = 2
	TABP_TABITEMRIGHTEDGE = 3	TIRES_SELECTED
		TIRES_DISABLED
		TIRES_FOCUSED
		TTIS_NORMAL = 1
		= 2 TTIS_SELECTE
	TABP_TOPTABITEM = 5	TTIS_DISABLED =
		TTIS_FOCUSED =
		TTIBES_NORMAL =
		TTIBES_HOT = 2
	TABP_TOPTABITEMBOTHEDGE = 8	TTIBES_SELECTE
		TTIBES_DISABLED
		TTIBES_FOCUSED
		TTILES_NORMAL =
		TTILES_HOT = 2
	TABP_TOPTABITEMLEFTEDGE = 6	TTILES_SELECTED
		TTILES_DISABLED
		TTILES_FOCUSED
		TTIRES_NORMAL
		TTIRES_HOT = 2
	TABP_TOPTABITEMRIGHTEDGE = 7	TTIRES_SELECTE
		TTIRES_DISABLED
		TTIRES_FOCUSED
TASKBAND	TDP_GROUPCOUNT = 1	
	TDP_FLASHBUTTON = 2	
	TDP_FLASHBUTTONGROUPMENU = 3	
TASKBAR	TBP_BACKGROUNDBOTTOM = 1	
	TBP_BACKGROUNDLEFT = 4	
	TBP_BACKGROUNDRIGHT = 2	

	TBP_BACKGROUNDTOP = 3	
	TBP_SIZINGBARBOTTOM = 5	
	TBP_SIZINGBARBOTTOMLEFT = 8	
	TBP_SIZINGBARRIGHT = 6	
	TBP_SIZINGBARTOP = 7	
TOOLBAR	TP_BUTTON = 1	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_DROPDOWNBUTTON = 2	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTON = 3	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTONDROPDOWN = 4	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATOR = 5	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATORVERT = 6	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
TOOLTIP	TTP_BALLOON = 3	TTBS_NORMAL = TTBS_LINK = 2
	TTP_BALLOONTITLE = 4	TTBS_NORMAL = TTBS_LINK = 2 TTCS_NORMAL =

	TTP_CLOSE = 5	TTCS_HOT = 2 TTCS_PRESSED =
	TTP_STANDARD = 1	TTSS_NORMAL = TTSS_LINK = 2
	TTP_STANDARDTITLE = 2	TTSS_NORMAL = TTSS_LINK = 2
TRACKBAR	TKP_THUMB = 3	TUS_NORMAL = 1 2 TUS_PRESSED = TUS_FOCUSED = TUS_DISABLED =
	TKP_THUMBBOTTOM = 4	TUBS_NORMAL = TUBS_HOT = 2 TUBS_PRESSED = TUBS_FOCUSED = TUBS_DISABLED =
	TKP_THUMBLEFT = 7	TUVLS_NORMAL = TUVLS_HOT = 2 TUVLS_PRESSED TUVLS_FOCUSED TUVLS_DISABLED
	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = TUVRS_HOT = 2 TUVRS_PRESSED TUVRS_FOCUSED TUVRS_DISABLED
	TKP_THUMBTOP = 5	TUTS_NORMAL = TUTS_HOT = 2 TUTS_PRESSED = TUTS_FOCUSED = TUTS_DISABLED =
	TKP_THUMBVERT = 6	TUVS_NORMAL = TUVS_HOT = 2 TUVS_PRESSED = TUVS_FOCUSED = TUVS_DISABLED =
	TKP_TICS = 9	TSS_NORMAL = 1
	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	TNP_ANIMBACKGROUND = 2	
	TNP_BACKGROUND = 1	

TREEVIEW	TVP_BRANCH = 3	
	TVP_GLYPH = 2	GLPS_CLOSED = GLPS_OPENED =
	TVP_TREEITEM = 1	TREIS_NORMAL = TREIS_HOT = 2 TREIS_SELECTED TREIS_DISABLED TREIS_SELECTED = 5
WINDOW	WP_CAPTION = 1	CS_ACTIVE = 1 CS = 2 CS_DISABLED
	WP_CAPTIONSIZINGTEMPLATE = 30	
	WP_CLOSEBUTTON = 18	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
	WP_DIALOG = 29	
	WP_FRAMEBOTTOM = 9	FS_ACTIVE = 1 FS = 2
	WP_FRAMEBOTTOMSIZINGTEMPLATE = 36	
	WP_FRAMELEFT = 7	FS_ACTIVE = 1 FS = 2
	WP_FRAMELEFTSIZINGTEMPLATE = 32	
	WP_FRAMERIGHT = 8	FS_ACTIVE = 1 FS = 2
	WP_FRAMERIGHTSIZINGTEMPLATE = 34	
	WP_HELPBUTTON = 23	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
	WP_HORZSCROLL = 25	HSS_NORMAL = 1 = 2 HSS_PUSHED HSS_DISABLED =
	WP_HORZTHUMB = 26	HTS_NORMAL = 1 2 HTS_PUSHED = HTS_DISABLED =
	WP_MAX_BUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED MXCS_ACTIVE = 1

WP_MAXCAPTION = 5	MXCS_INACTIVE = MXCS_DISABLED
WP_MDICLOSEBUTTON = 20	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_MDIHELPBUTTON = 24	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_MDIMINBUTTON = 16	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
WP_MDIRESTOREBUTTON = 22	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_MDISYSBUTTON = 14	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_MINBUTTON = 15	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
WP_MINCAPTION = 3	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_RESTOREBUTTON = 21	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLCAPTION = 2	CS_ACTIVE = 1 CS = 2 CS_DISABLED
WP_SMALLCAPTIONSIZINGTEMPLATE = 31	
WP_SMALLCLOSEBUTTON = 19	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_SMALLFRAMEBOTTOM = 12	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37	
WP_SMALLFRAMELEFT = 10	FS_ACTIVE = 1 FS = 2

WP_SMALLFRAMELEFTSIZINGTEMPLATE = 33	
WP_SMALLFRAMERIGHT = 11	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35	
WP_SMALLHELPBUTTON	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_VERTSCROLL = 27	VSS_NORMAL = 1 = 2 VSS_PUSHED VSS_DISABLED =
WP_VERTTHUMB = 28	VTS_NORMAL = 1 2 VTS_PUSHED = VTS_DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

Use the Clear method to clear all skins from the control. Use the Remove method to remove a specific skin. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The skin method may change the visual appearance for the following parts in the control:

- control's border, Appearance property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- **selected item** or cell, SelBackColor property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltip, and so on, Background property

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

Type	Description
ID as Long	A Long expression that indicates the index of the skin being removed.

Use the Remove method to remove a specific skin. The identifier of the skin being removed should be the same as when the skin was added using the Add method. Use the Clear method to clear all skins from the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The skin method may change the visual appearance for the following parts in the control:

- control's border, Appearance property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- **selected item** or cell, SelBackColor property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltip, and so on, Background property

property Appearance.RenderType as Long

Specifies the way colored EBN objects are displayed on the component.

Type	Description
Long	A long expression that indicates how the EBN objects are shown in the control, like explained bellow.

By default, the RenderType property is 0, which indicates an A-color scheme. The RenderType property can be used to change the colors for the entire control, for parts of the controls that uses EBN objects. The RenderType property is not applied to the currently XP-theme if using.

The RenderType property is applied to all parts that displays an EBN object. The properties of color type may support the EBN object if the property's description includes "*A color expression that indicates the cell's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.*" In other words, a property that supports EBN objects should be of format 0xIDRRGGBB, where the ID is the identifier of the EBN to be applied, while the BBGGRR is the (Red,Green,Blue, RGB-Color) color to be applied on the selected EBN. For instance, the 0x1000000 indicates displaying the EBN as it is, with no color applied, while the 0x1FF0000, applies the Blue color (RGB(0x0,0x0,0xFF), RGB(0,0,255) on the EBN with the identifier 1. You can use the EBNColor tool to visualize applying EBN colors.

Click here to watch a movie on how you can change the colors to be applied on EBN objects.

For instance, the following sample changes the control's header appearance, by using an EBN object:

```
With Control
   .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
   .BackColorHeader = &H1000000
End With
```

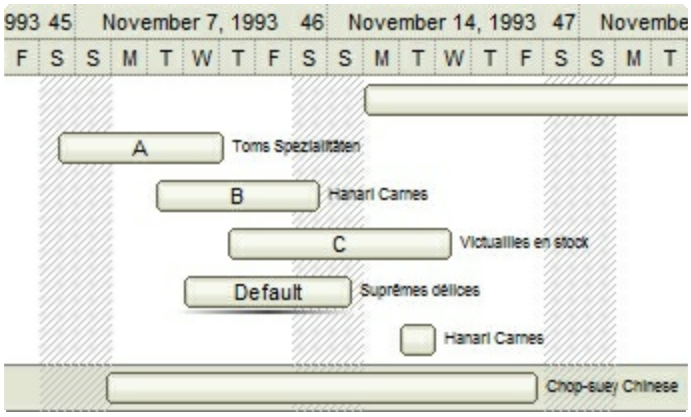
In the following screen shot the following objects displays the current EBN with a different color:

- "A" in Red (RGB(255,0,0), for instance the bar's property exBarColor is 0x10000FF
- "B" in Green (RGB(0,255,0), for instance the bar's property exBarColor is 0x100FF00

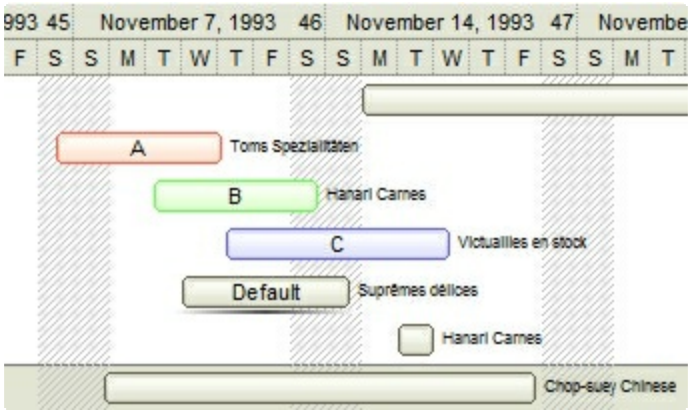
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

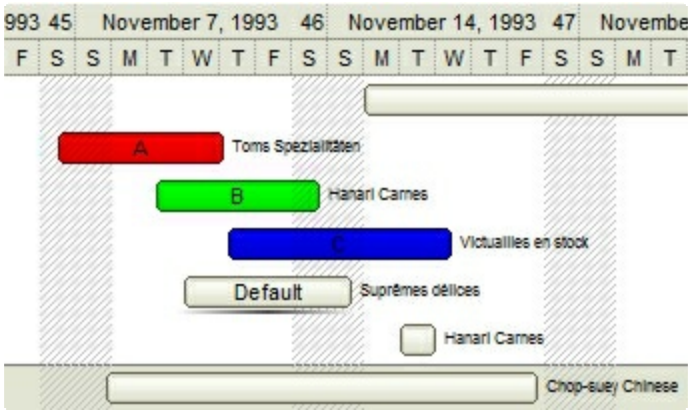
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



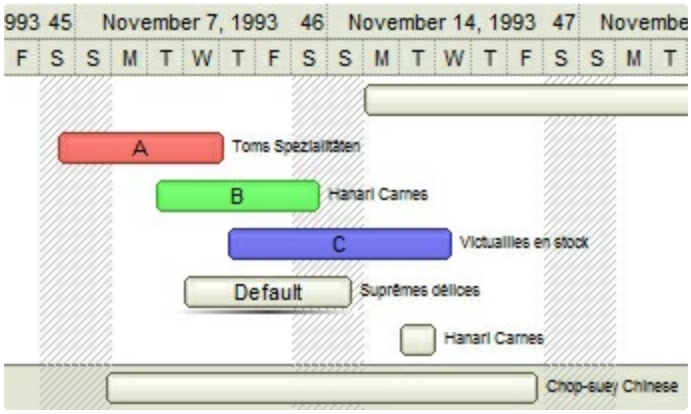
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the part of the control. This option should be used with solid colors (RGB(255,0,0), RGB(0,255,0), RGB(0,0,255), RGB(255,255,0), RGB(255,0,255), RGB(0,255,255),RGB(127,0,0),RGB(0,127,0), ...)



- ***-1**, AND-color scheme*, The color to be applied on the part of the control is an AND bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the AND bit for the entire Blue channel, or in other words, it applies a more Blue to the part of the control. This option should be used with solid colors (RGB(255,0,0), RGB(0,255,0), RGB(0,0,255), RGB(255,255,0), RGB(255,0,255), RGB(0,255,255),RGB(127,0,0),RGB(0,127,0), ...)

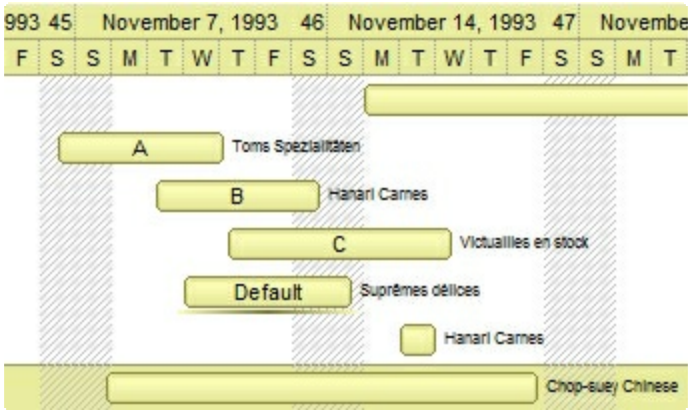


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

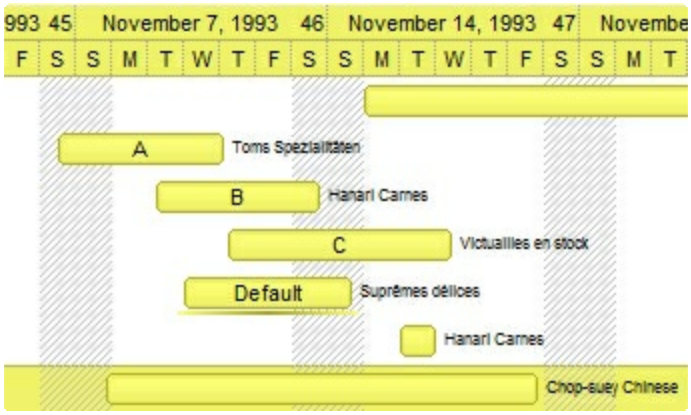


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on 0x4000FFFF, indicates a 25% Yellow on EBN objects. The 0x40, or 64 in decimal, is a 25 % from in a 256 interal, and the 0x00FFFF, indicates the Yellow (RGB(255,255,0)). The same could be if the RenderType is 0x40000000 + vbYellow, or &H40000000 + RGB(255, 255, 0), and so, the RenderType could be the 0xAA000000 + Color, where the Color is the RGB format of the color.

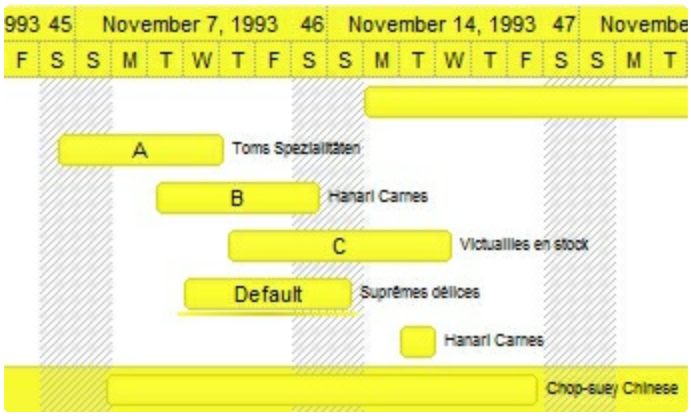
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



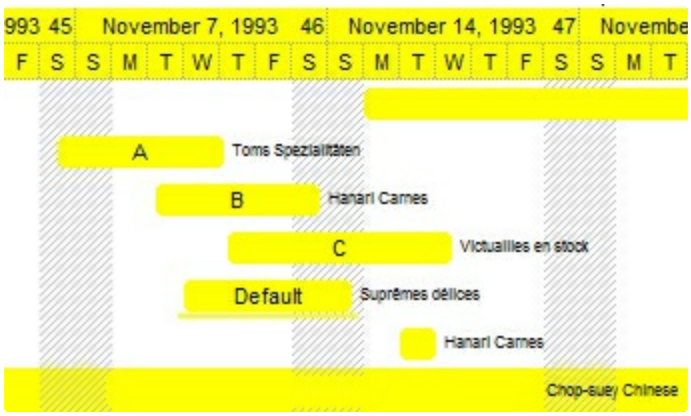
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



*The following picture shows the control with the RenderType property on 0x**C0**00FFFF (75% Yellow, 0xC0 or 192 in decimal is 75% from 256):*



*The following picture shows the control with the RenderType property on 0x**FF**00FFFF (100% Yellow, 0xFF or 255 in decimal is 100% from 255):*


993 45
November 7, 1993
46
November 14, 1993
47
Novembe
F S S M T W T F S S M T W T F S S M T
A
Toms Spezialitäten
B
Hanari Carnes
C
Victuailles en stock
Default
Suprêmes délices
Hanari Carnes
Chop-suey Chinese

Column object

A Column object holds information about a data column. The DataColumns property gives access to the Column objects.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method

The Column object supports the following properties and methods:

Name	Description
Alignment	Specifies the column's alignment.
AllowGroupBy	Specifies whether the user can group by this column.
Caption	Retrieves or sets the text displayed to the column's header.
DefaultFormatAppearances	Specifies the list of format-appearances (key of FormatAppearance object), separated by comma, to be applied on the column when it is displayed in the pivot-table.
DefaultFormatContent	Specifies the default format (key of FormatContent object) to be applied on the column when it is displayed in the pivot-table.
FormatImage	Defines the expression to determine the images the column display.
HeaderAlignment	Specifies the alignment of the column's caption.
HTML	Indicates whether the column handles/displays built-in HTML format.
Index	Returns index of the object within the collection.
PivotCaption	Specifies the caption of the column to be displayed on the columns floating toolbox.
SortOrder	Specifies the sorting order of the pivot column when it is dropped to the control's pivot bar.
SortType	Returns or sets a value that indicates the way a control sorts the values for a column.

property Column.Alignment as AlignmentEnum

Specifies the column's alignment.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the column's content.

By default, the Alignment property is LeftAlignment. Use the Alignment property to change the column's alignment. The column's alignment is keep once you drag or group by that column. The HeaderAlignment property specifies the alignment of the caption in the column's header.

property Column.AllowGroupBy as AllowGroupByEnum

Specifies whether the user can group by this column.

Type	Description
AllowGroupByEnum	An AllowGroupByEnum expression that specifies where the column can be dropped on the control's pivot bar.

By default, the AllowGroupBy property is exGroupByAny, which indicates that the column can be dropped anywhere on the control's pivot bar. The PivotBarVisible property shows or hides the control's pivot bar. Use the AllowGroupBy property to prevent the column to group by that column. For instance, you can prevent grouping by numeric columns. Use the AllowDrop property on False, to prevent loading the data-files (TXT, XML files), by drag and drop, into the control. The AllowGroupBy property has effect for columns dragged from the control's Columns Floating Panel (PivotColumnsFloatBarVisible property), or from the control's columns header.

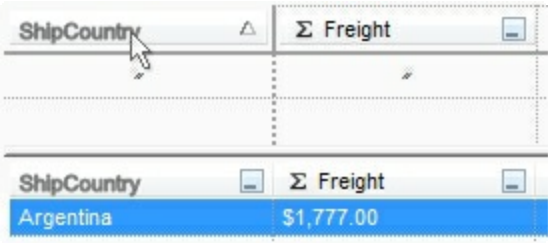
property Column.Caption as String

Retrieves or sets the text displayed to the column's header.

Type	Description
String	A String expression that specifies the column's caption to be displayed on the control' headers.

By default, the Caption property is initialized with the name of the column as it is loaded. The Caption property supports built-HTML tags, so you can include icons or images using the <img> tags. The PivotCaption property specifies the caption of the column to be displayed on the floating columns panel. The SortType property indicates the sorting type for the column, so it indicates the type of filter being shown. The FormatPivotHeader property indicates the format of the caption to be displayed on the headers. The **caption** keyword in the FormatPivotHeader property indicates the Caption property of the Column object.

The following screen shot shows where the Caption goes:



The Caption property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with

a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the

text such as: Text with subscript

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

property Column.DefaultFormatAppearances as String

Specifies the list of format-appearances (key of FormatAppearance object), separated by comma, to be applied on the column when it is displayed in the pivot-table.

Type	Description
String	A String expression that specifies the list of format-appearances (key of FormatAppearance object), separated by comma, to be applied on the column when it is displayed in the pivot-table.

By default, the DefaultFormatAppearances property is empty. The DefaultFormatAppearances property is not applied when the control displays initially the data. The DefaultFormatAppearances is applied only when the column is dragged from the control's columns panel (PivotColumnsFloatBarVisible property). The PivotColumnsFloatBarVisible property retrieves or sets a value that indicates whether the pivot columns float bar is visible or hidden. The DefaultFormatContent property specifies the default format (key of FormatContent object) to be applied on the column when it is displayed in the pivot-table. The DefaultFormatAppearances property can include any Key on the FormatAppearances collection. For instance, "*bold,italic*" specifies that the column will be displayed in bold and italic. The PivotColumns property specifies the list of columns that shows the summarized data.

property Column.DefaultFormatContent as String

Specifies the default format (key of FormatContent object) to be applied on the column when it is displayed in the pivot-table.

Type	Description
String	A String expression that specifies the default format (key of FormatContent object) to be applied on the column when it is displayed in the pivot-table.

By default, the DefaultFormatContent property is empty. The DefaultFormatContent property specifies the default format (key of FormatContent object) to be applied on the column when it is displayed in the pivot-table. The DefaultFormatContent property is not applied when the control displays initially the data. The DefaultFormatContent is applied only when the column is dragged from the control's columns panel (PivotColumnsFloatBarVisible property). The PivotColumnsFloatBarVisible property retrieves or sets a value that indicates whether the pivot columns float bar is visible or hidden. The DefaultFormatContent property can include any Key on the FormatContents collection. For instance, "*numeric*" specifies that the column will be displayed as numeric. The PivotColumns property specifies the list of columns that shows the summarized data.

property Column.FormatImage as String

Defines the expression to determine the images the column display.

Type	Description
String	A string value that defines the expression that determines the icons to display within the column, based on the values.

By default, the FormatImage property is empty (it indicates that no icons is being displayed). The FormatImage property defines the expression to determine the images the column display. Use the Images method to insert icons at runtime. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The FormatImage expression should return an integer value that specifies the index of the icon to displayed (zero indicates that the column displays no icons). The expression may be a combination of variables, constants, strings, dates and operators, and **value**. The ***value*** operator gives the value of the cell to display the icon.

For instance:

- "" or "0", no icons is being shown within the column
- "*1*", displays the icon with the index 1 (first icon) for the entire column
- "*"1 pos `` mod 2 ? 1 : -1"*" , displays alternate icons
- "*%1 = 'NM' ? 1 : 0*", display the first icon for all values NM within the column with the index 1
- "*value case (default: -1;'Germany': 1;'USA': 2;'Mexico': 3)*", displays the icon with the index 1 for Germany, the icon with the index 2 for USA, the icon with the index 3 for Mexico, and empty icon for every other value.

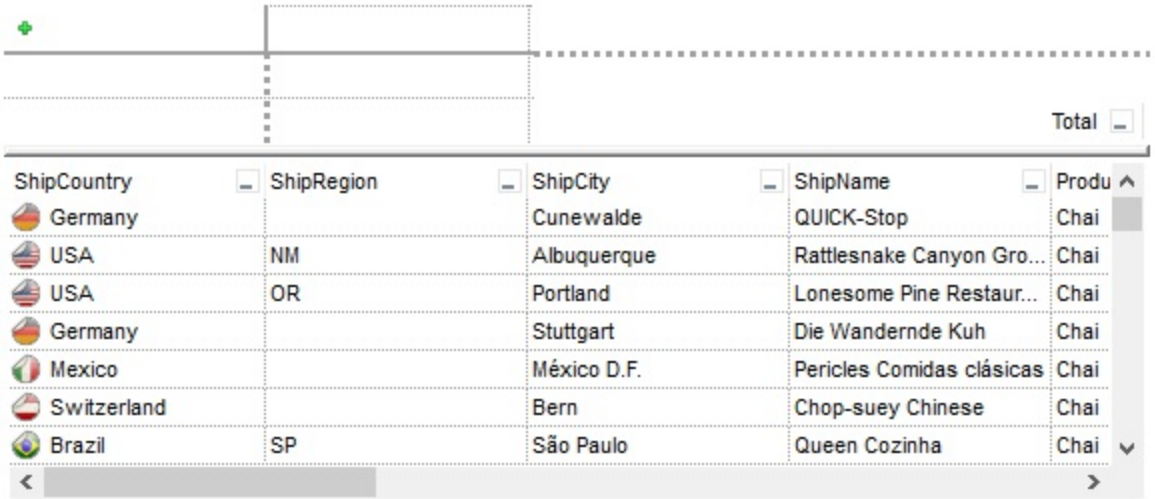
The **value** keyword in the FormatColumn property indicates the value of the cell.

The expression supports cell's identifiers as follows:

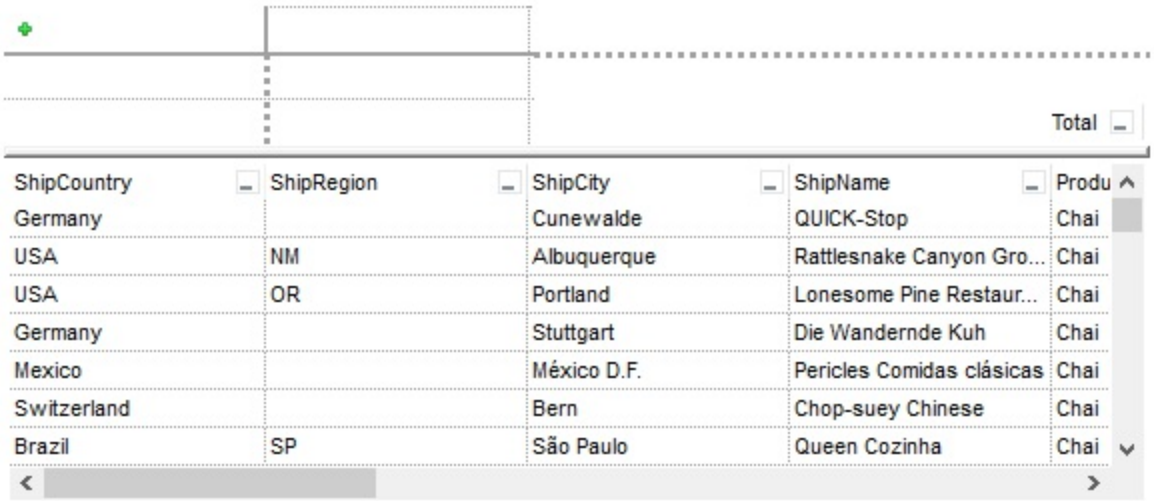
- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ...*

This property/method supports predefined constants and operators/functions as described here.

The following screen shot shows the control with icons:



The following screen shot shows the control with no icons:



property Column.HeaderAlignment as AlignmentEnum

Specifies the alignment of the column's caption.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the column's caption.

By default, the HeaderAlignment property is LeftAlignment. Use the HeaderAlignment property to change the column's caption alignment. The column's alignment is keep once you drag or group by that column. The Alignment property specifies the alignment of the column's content (the cells).

property Column.HTML as Boolean

Indicates whether the column handles/displays built-in HTML format.

Type	Description
Boolean	A Boolean expression that specifies whether the column built-in HTML format.

By default, the HTML property is False, which indicates that the column display plain-text, even if it contains HTML tags. Use the HTML property to specify whether the column includes/displays HTML tags.

The built-in HTML tags supported are:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray,

width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

property Column.Index as Long

Returns index of the object within the collection.

Type	Description
Long	A Long expression that specifies the index of the Column object in the data Columns collection.

The Index property is 0-based, so the first added column has 0 as index. The Index property is read-only, and it is assigned by the control once the data is loaded to the control.

property Column.PivotCaption as String

Specifies the caption of the column to be displayed on the columns floating toolbox.

Type	Description
String	A String expression that specifies the column's caption to be displayed on the control' headers.

By default, the PivotCaption property is initialized with the name of the column as it is loaded (if the data provided contains no header information, the PivotCaption may display the name such as Column 1 (value)). The PivotColumnsFloatBarVisible property specifies whether the Columns collection is displayed to a floating bar, so user can drag and drop columns to the control's pivot bar so it gets data summarized. The PivotCaption property specifies the caption of the column to be displayed on the floating columns panel. The PivotCaption property supports built-HTML tags, so you can include icons or images using the <img> tags. The SortType property indicates the sorting type for the column, so it indicates the type of filter being shown. The Caption property specifies the caption to be displayed on the column's header.

The following screen shot shows where the PivotCaption goes:

The PivotCaption property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using

the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.

- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the

offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

  gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

  outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

  shadow

  or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

  outline anti-aliasing

property Column.SortOrder as SortOrderEnum

Specifies the sorting order of the pivot column when it is dropped to the control's pivot bar.

Type	Description
SortOrderEnum	A SortOrderEnum expression that specified the default sorting order of the column.

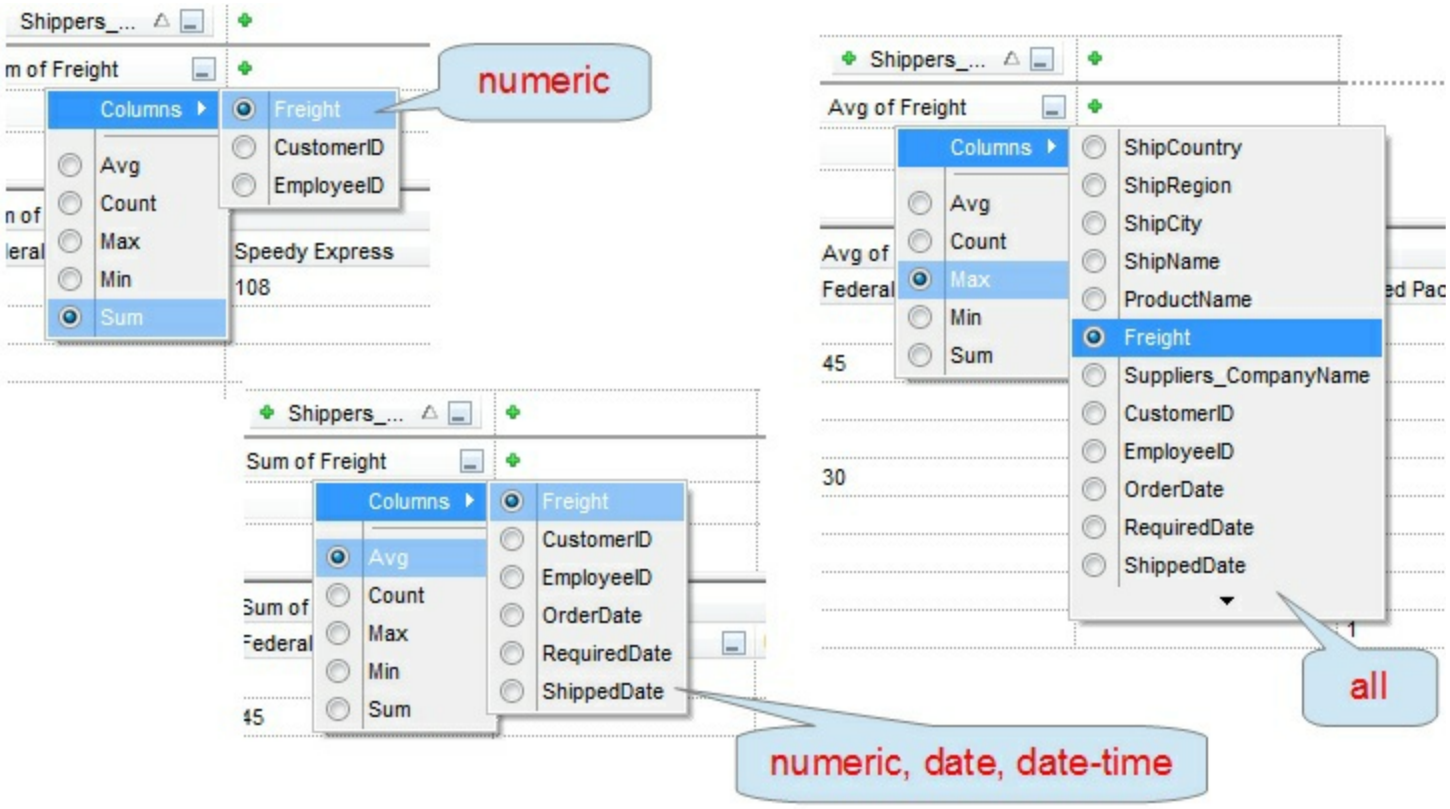
By default, the SortOrder property is SortNone. Use the SortOrder property to specify the default sorting order of the pivot column. It specifies the sorting order of the pivot column when inserted to the control's pivot bar by drag and drop or when selecting from the + (expand button) on the control's pivot bar. The SortType property returns or sets a value that indicates the way a control sorts the values for a column. Use the DisplayFilterList property to specify whether the column's header displays a drop down filter button.

property Column.SortType as SortTypeEnum

Returns or sets a value that indicates the way a control sorts the values for a column.

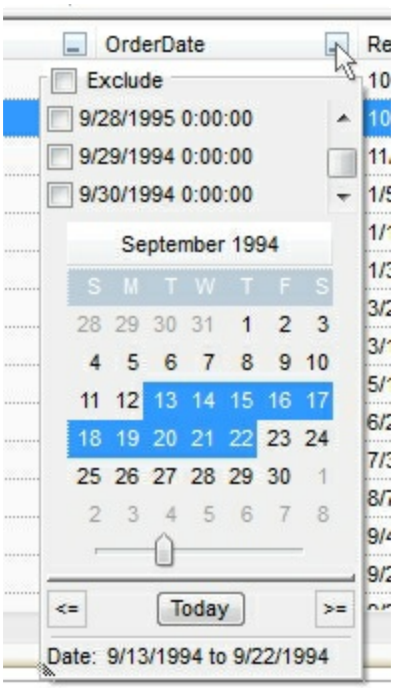
Type	Description
SortTypeEnum	A SortTypeEnum expression that determines the sorting type.

By default, the SortType property is determined by the type of the column being loaded. Use the SortOrder property to specify the default sorting order of the pivot column. For instance, if data is loaded using the DataSource, the Type of the Column/Field determines the sorting type. Based on the SortType property, the control displays the associated filtering box. Use the DisplayFilterList property to specify whether the column's header displays a drop down filter button. You can use the FilterBarPromptVisible property to show or hide the control's filter prompt field. Based on the column's SortType property, the aggregate functions displays the associated columns as shown in the following screen shot:



The following screen shot shows the filtering box, if the SortType property is SortString:

The following screen shot shows the filtering box, if the SortType property is SortDate:



Columns object

The Columns objects holds a collection of Column objects. The DataColumns property gives access to the Column objects.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method

The Columns object supports the following properties and methods:

Name	Description
Count	Returns the number of objects in a collection.
Item	Returns a specific Column giving its index/caption.

property Columns.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A Long expression that specifies the number of data columns

The Count property specifies the number of data columns. The ClearData method clears the control's data. The PivotColumnsFloatBarVisible property specifies whether the Columns collection is displayed to a floating bar, so user can drag and drop columns to the control's pivot bar so it gets data summarized. You can use the **for each** statement to enumerate the data columns in the control. The Item property gets the Column object based on its index or caption.

The following VB sample enumerates the data columns:

```
With Pivot1
   Dim c As EXPIVOTLibCtl.Column
   For Each c In .DataColumns
      Debug.Print c.Caption
   Next
End With
```

property Columns.Item (Index as Variant) as Column

Returns a specific Column giving its index/caption.

Type	Description
Index as Variant	A Long expression that specifies the index of the column to be requested, a String expression that indicates the caption of the column to be accessed.
Column	A Column object being requested.

The Item property gets the Column object based on its index or caption. The Count property specifies the number of data columns. The ClearData method clears the control's data. The PivotColumnsFloatBarVisible property specifies whether the Columns collection is displayed to a floating bar, so user can drag and drop columns to the control's pivot bar so it gets data summarized. You can use the **for each** statement to enumerate the data columns in the control.

The following VB sample enumerates the data columns:

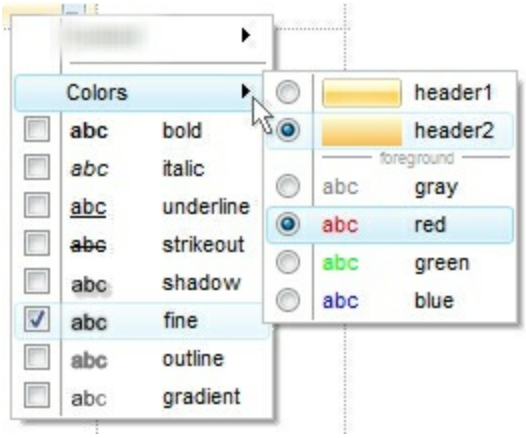
```
With Pivot1
   Dim c As EXPIVOTLibCtl.Column
   For Each c In .DataColumns
      Debug.Print c.Caption
   Next
End With
```

FormatAppearance object

The FormatAppearance objects can be accessed through the control's FormatAppearances property. The FormatAppearance object changes the visual appearance of your data as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

The following screen shot shows the control's context menu with the FormatAppearance objects:



The FormatAppearance object supports the following properties and methods:

Name	Description
BackColor	Specifies the element's background color.
Bold	Renders as bold text.
Font	Retrieves or sets the text's font.
FontSize	Indicates the size of the font to display the text.
ForeColor	Specifies the element's foreground color.
Gradient	Renders the text with a gradient color.
GradientMode	Indicates the gradient mode to be applied on text.
Italic	Renders as italic text.
Key	Indicates the key of the FormatAppearance object.
Name	Specifies the name of the FormatAppearance object to be displayed on the context menu.
Outline	Renders the text outlined with specified color.
OutlineSize	Indicates the size of the outline to be applied on text.

Shadow	Renders the text with a shadow of specified color.
ShadowOffset	Indicates the offset of the shadow to be applied on text.
ShadowSize	Indicates the size of the shadow to be applied on text.
StrikeOut	Specifies that the text should appear as strikeout.
ToolTip	Specifies the tooltip of the FormatAppearance object to be displayed when the cursor hovers the object.
Underline	Underlines the text.

property FormatAppearance.BackColor as Color

Specifies the element's background color.

Type	Description
Color	A color expression that defines the background color to be applied. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The BackColor property defines the background color to be applied. This property supports solid colors or EBN colors. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part. The ForeColor property defines the foreground color to be applied on the objects (columns/rows).

- How can I display the total with a solid background color?
- How can I display the total with a different background color/ebn?

How can I display the total with a solid background color?

VBA (MS Access, Excell...)

```
With Pivot1
   .FormatAppearances.Add("back").BackColor = RGB(240,240,240)
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[back]"
End With
```

VB6

```
With Pivot1
   .FormatAppearances.Add("back").BackColor = RGB(240,240,240)
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
```

```
    .PivotRows = "0"
    .PivotColumns = "sum(5)"
    .PivotTotals = "sum[back]"
End With
```

VB.NET

```
With Expivot1
    .FormatAppearances.Add("back").BackColor = Color.FromArgb(240,240,240)
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .PivotRows = "0"
    .PivotColumns = "sum(5)"
    .PivotTotals = "sum[back]"
End With
```

VB.NET for /COM

```
With AxPivot1
    .FormatAppearances.Add("back").BackColor = RGB(240,240,240)
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .PivotRows = "0"
    .PivotColumns = "sum(5)"
    .PivotTotals = "sum[back]"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->GetFormatAppearances()->Add(L"back",vtMissing)-
>PutBackColor(RGB(240,240,240));
spPivot1->Import("C:\\Program
```

```
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->PutPivotRows(L"0");
spPivot1->PutPivotColumns(L"sum(5)");
spPivot1->PutPivotTotals(L"sum[back]");
```

C++ Builder

```
Pivot1->FormatAppearances->Add(L"back",TNoParam())->BackColor =
RGB(240,240,240);
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->PivotRows = L"0";
Pivot1->PivotColumns = L"sum(5)";
Pivot1->PivotTotals = L"sum[back]";
```

C#

```
expivot1.FormatAppearances.Add("back",null).BackColor =
Color.FromArgb(240,240,240);
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.PivotRows = "0";
expivot1.PivotColumns = "sum(5)";
expivot1.PivotTotals = "sum[back]";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
  Pivot1.FormatAppearances.Add("back",null).BackColor = 15790320;
  Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
  Pivot1.PivotRows = "0";
  Pivot1.PivotColumns = "sum(5)";
  Pivot1.PivotTotals = "sum[back]";
```

```
</SCRIPT>
```

C# for /COM

```
axPivot1.FormatAppearances.Add("back",null).BackColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(240,240,240));
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.PivotRows = "0";
axPivot1.PivotColumns = "sum(5)";
axPivot1.PivotTotals = "sum[back]";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatAppearance;
   anytype var_FormatAppearance;
   ;

   super();

   var_FormatAppearance =
COM::createFromObject(expivot1.FormatAppearances()).Add("back");
com_FormatAppearance = var_FormatAppearance;
   com_FormatAppearance.BackColor(WinApi::RGB2int(240,240,240));
   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   expivot1.PivotRows("0");
   expivot1.PivotColumns("sum(5)");
   expivot1.PivotTotals("sum[back]");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   FormatAppearances.Add('back',Nil).BackColor := $f0f0f0;
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
```

```
   PivotRows := '0';
   PivotColumns := 'sum(5)';
   PivotTotals := 'sum[back]';
end
```

Delphi (standard)

```
with Pivot1 do
begin
   FormatAppearances.Add('back',Null).BackColor := $f0f0f0;
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
   PivotRows := '0';
   PivotColumns := 'sum(5)';
   PivotTotals := 'sum[back]';
end
```

VFP

```
with thisform.Pivot1
   .FormatAppearances.Add("back").BackColor = RGB(240,240,240)
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[back]"
endwith
```

dBASE Plus

```
local oPivot,var_FormatAppearance

oPivot = form.Activex1.nativeObject
// oPivot.FormatAppearances.Add("back").BackColor = 0xf0f0f0
var_FormatAppearance = oPivot.FormatAppearances.Add("back")
with (oPivot)
   TemplateDef = [Dim var_FormatAppearance]
   TemplateDef = var_FormatAppearance
   Template = [var_FormatAppearance.BackColor = 0xf0f0f0]
endwith
```

```
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[back]"
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
' oPivot.FormatAppearances.Add("back").BackColor = 15790320
var_FormatAppearance = oPivot.FormatAppearances.Add("back")
oPivot.TemplateDef = "Dim var_FormatAppearance"
oPivot.TemplateDef = var_FormatAppearance
oPivot.Template = "var_FormatAppearance.BackColor = 15790320"

oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[back]"
```

Visual Objects

```
oDCOCX_Exontrol1:FormatAppearances:Add("back",nil):BackColor :=
RGB(240,240,240)
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:PivotRows := "0"
oDCOCX_Exontrol1:PivotColumns := "sum(5)"
oDCOCX_Exontrol1:PivotTotals := "sum[back]"
```

PowerBuilder

```
OleObject oPivot
```

```
oPivot = ole_1.Object
oPivot.FormatAppearances.Add("back").BackColor = RGB(240,240,240)
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[back]"
```

How can I display the total with a different background color/ebn?

VBA (MS Access, Excell...)

```
With Pivot1
   .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
   .FormatAppearances.Add("back").BackColor = &H1000000
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[back]"
End With
```

VB6

```
With Pivot1
   .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
   .FormatAppearances.Add("back").BackColor = &H1000000
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[back]"
End With
```

VB.NET

```
With Expivot1
   .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
   .FormatAppearances.Add("back").BackColor32 = &H1000000
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
```

```
    .PivotColumns = "sum(5)"
    .PivotTotals = "sum[back]"
End With
```

VB.NET for /COM

```
With AxPivot1
    .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
    .FormatAppearances.Add("back").BackColor = &H1000000
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .PivotRows = "0"
    .PivotColumns = "sum(5)"
    .PivotTotals = "sum[back]"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->GetVisualAppearance()->Add(1,"c:\\exontrol\\images\\normal.ebn");
spPivot1->GetFormatAppearances()->Add(L"back",vtMissing)-
>PutBackColor(0x1000000);
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->PutPivotRows(L"0");
spPivot1->PutPivotColumns(L"sum(5)");
spPivot1->PutPivotTotals(L"sum[back]");
```

C++ Builder

```
Pivot1->VisualAppearance->Add(1,TVariant("c:\\exontrol\\images\\normal.ebn"));
```

```
Pivot1->FormatAppearances->Add(L"back",TNoParam())->BackColor = 0x1000000;
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->PivotRows = L"0";
Pivot1->PivotColumns = L"sum(5)";
Pivot1->PivotTotals = L"sum[back]";
```

C#

```
expivot1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
expivot1.FormatAppearances.Add("back",null).BackColor32 = 0x1000000;
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.PivotRows = "0";
expivot1.PivotColumns = "sum(5)";
expivot1.PivotTotals = "sum[back]";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
   Pivot1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
   Pivot1.FormatAppearances.Add("back",null).BackColor = 16777216;
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   Pivot1.PivotRows = "0";
   Pivot1.PivotColumns = "sum(5)";
   Pivot1.PivotTotals = "sum[back]";
</SCRIPT>
```

C# for /COM

```
axPivot1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
axPivot1.FormatAppearances.Add("back",null).BackColor = 0x1000000;
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.PivotRows = "0";
```

```
axPivot1.PivotColumns = "sum(5)";
axPivot1.PivotTotals = "sum[back]";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatAppearance;
   anytype var_FormatAppearance;
   ;

   super();

   expivot1.VisualAppearance().Add(1,"c:\\exontrol\\images\\normal.ebn");
   var_FormatAppearance =
COM::createFromObject(expivot1.FormatAppearances()).Add("back");
com_FormatAppearance = var_FormatAppearance;
   com_FormatAppearance.BackColor(0x1000000);
   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   expivot1.PivotRows("0");
   expivot1.PivotColumns("sum(5)");
   expivot1.PivotTotals("sum[back]");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   VisualAppearance.Add(1,'c:\exontrol\images\normal.ebn');
   FormatAppearances.Add('back',Nil).BackColor := $1000000;
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   PivotRows := '0';
   PivotColumns := 'sum(5)';
   PivotTotals := 'sum[back]';
end
```

Delphi (standard)

```
with Pivot1 do
begin
   VisualAppearance.Add(1,'c:\exontrol\images\normal.ebn');
   FormatAppearances.Add('back',Null).BackColor := $1000000;
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
   PivotRows := '0';
   PivotColumns := 'sum(5)';
   PivotTotals := 'sum[back]';
end
```

VFP

```
with thisform.Pivot1
   .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
   .FormatAppearances.Add("back").BackColor = 0x1000000
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[back]"
endwith
```

dBASE Plus

```
local oPivot,var_FormatAppearance

oPivot = form.Activex1.nativeObject
oPivot.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
// oPivot.FormatAppearances.Add("back").BackColor = 0x1000000
var_FormatAppearance = oPivot.FormatAppearances.Add("back")
with (oPivot)
   TemplateDef = [Dim var_FormatAppearance]
   TemplateDef = var_FormatAppearance
   Template = [var_FormatAppearance.BackColor = 0x1000000]
endwith
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[back]"
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
' oPivot.FormatAppearances.Add("back").BackColor = 16777216
var_FormatAppearance = oPivot.FormatAppearances.Add("back")
oPivot.TemplateDef = "Dim var_FormatAppearance"
oPivot.TemplateDef = var_FormatAppearance
oPivot.Template = "var_FormatAppearance.BackColor = 16777216"

oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[back]"
```

Visual Objects

```
oDCOCX_Exontrol1:VisualAppearance:Add(1,"c:\exontrol\images\normal.ebn")
oDCOCX_Exontrol1:FormatAppearances:Add("back",nil):BackColor := 0x1000000
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:PivotRows := "0"
oDCOCX_Exontrol1:PivotColumns := "sum(5)"
oDCOCX_Exontrol1:PivotTotals := "sum[back]"
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
```

```
oPivot.FormatAppearances.Add("back").BackColor = 16777216 /*0x1000000*/
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[back]"
```

property FormatAppearance.Bold as Boolean

Renders as bold text.

Type	Description
Boolean	A Boolean expression that specifies whether the text is rendered in bold.

By default, the Bold property is False. Use the Bold property on True, to show the objects in **bold**.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.Font as IFontDisp

Retrieves or sets the text's font.

Type	Description
IFontDisp	A Font object that indicates the font to be applied by the FormatAppearance object.

By default, the Font property is nothing/empty, which indicates no effect. You can use the Font property to specify a different font to be applied on selected columns/rows. The FontSize property can be used to change just the size of the current font, which is applied to the current selection.

property FormatAppearance.FontSize as Long

Indicates the size of the font to display the text.

Type	Description
Long	A Long expression that defines the size of the font

By default, the FontSize property is 0, which indicates no effect. The FontSize property can be used to change just the size of the current font, which is applied to the current selection. You can use the Font property to specify a different font to be applied on selected columns/rows.

property FormatAppearance.ForeColor as Color

Specifies the element's foreground color.

Type	Description
Color	A Color expression that defines the color to be applied on the object's foreground.

The ForeColor property specifies the foreground color to be applied on the columns/rows. The BackColor property defines the background color to be applied on the objects (columns/rows).

How can I display the total with a different foreground color?

VBA (MS Access, Excell...)

```
With Pivot1
   .FormatAppearances.Add("fore").ForeColor = RGB(255,0,0)
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[fore,bold]"
End With
```

VB6

```
With Pivot1
   .FormatAppearances.Add("fore").ForeColor = RGB(255,0,0)
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[fore,bold]"
End With
```

VB.NET

```
With Expivot1
   .FormatAppearances.Add("fore").ForeColor = Color.FromArgb(255,0,0)
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
```

```
    .PivotTotals = "sum[fore,bold]"
End With
```

VB.NET for /COM

```
With AxPivot1
    .FormatAppearances.Add("fore").ForeColor = RGB(255,0,0)
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .PivotRows = "0"
    .PivotColumns = "sum(5)"
    .PivotTotals = "sum[fore,bold]"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->GetFormatAppearances()->Add(L"fore",vtMissing)-
>PutForeColor(RGB(255,0,0));
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->PutPivotRows(L"0");
spPivot1->PutPivotColumns(L"sum(5)");
spPivot1->PutPivotTotals(L"sum[fore,bold]");
```

C++ Builder

```
Pivot1->FormatAppearances->Add(L"fore",TNoParam())->ForeColor =
RGB(255,0,0);
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
```

```
Pivot1->PivotRows = L"0";
Pivot1->PivotColumns = L"sum(5)";
Pivot1->PivotTotals = L"sum[fore,bold]";
```

C#

```
expivot1.FormatAppearances.Add("fore",null).ForeColor = Color.FromArgb(255,0,0);
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.PivotRows = "0";
expivot1.PivotColumns = "sum(5)";
expivot1.PivotTotals = "sum[fore,bold]";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
  Pivot1.FormatAppearances.Add("fore",null).ForeColor = 255;
  Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
  Pivot1.PivotRows = "0";
  Pivot1.PivotColumns = "sum(5)";
  Pivot1.PivotTotals = "sum[fore,bold]";
</SCRIPT>
```

C# for /COM

```
axPivot1.FormatAppearances.Add("fore",null).ForeColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0));
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.PivotRows = "0";
axPivot1.PivotColumns = "sum(5)";
axPivot1.PivotTotals = "sum[fore,bold]";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_FormatAppearance;
    anytype var_FormatAppearance;
    ;

    super();

    var_FormatAppearance =
COM::createFromObject(expivot1.FormatAppearances()).Add("fore");
com_FormatAppearance = var_FormatAppearance;
    com_FormatAppearance.ForeColor(WinApi::RGB2int(255,0,0));
    expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
    expivot1.PivotRows("0");
    expivot1.PivotColumns("sum(5)");
    expivot1.PivotTotals("sum[fore,bold]");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
    FormatAppearances.Add('fore',Nil).ForeColor := $ff;
    Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
    PivotRows := '0';
    PivotColumns := 'sum(5)';
    PivotTotals := 'sum[fore,bold]';
end
```

Delphi (standard)

```
with Pivot1 do
begin
    FormatAppearances.Add('fore',Null).ForeColor := $ff;
    Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
    PivotRows := '0';
    PivotColumns := 'sum(5)';
    PivotTotals := 'sum[fore,bold]';
```

```
end
```

VFP

```
with thisform.Pivot1
   .FormatAppearances.Add("fore").ForeColor = RGB(255,0,0)
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
   .PivotColumns = "sum(5)"
   .PivotTotals = "sum[fore,bold]"
endwith
```

dBASE Plus

```
local oPivot,var_FormatAppearance

oPivot = form.Activex1.nativeObject
// oPivot.FormatAppearances.Add("fore").ForeColor = 0xff
var_FormatAppearance = oPivot.FormatAppearances.Add("fore")
with (oPivot)
   TemplateDef = [Dim var_FormatAppearance]
   TemplateDef = var_FormatAppearance
   Template = [var_FormatAppearance.ForeColor = 0xff]
endwith
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[fore,bold]"
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
' oPivot.FormatAppearances.Add("fore").ForeColor = 255
var_FormatAppearance = oPivot.FormatAppearances.Add("fore")
```

```
oPivot.TemplateDef = "Dim var_FormatAppearance"
oPivot.TemplateDef = var_FormatAppearance
oPivot.Template = "var_FormatAppearance.ForeColor = 255"

oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[fore,bold]"
```

Visual Objects

```
oDCOCX_Exontrol1:FormatAppearances:Add("fore",nil):ForeColor := RGB(255,0,0)
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:PivotRows := "0"
oDCOCX_Exontrol1:PivotColumns := "sum(5)"
oDCOCX_Exontrol1:PivotTotals := "sum[fore,bold]"
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.FormatAppearances.Add("fore").ForeColor = RGB(255,0,0)
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)"
oPivot.PivotTotals = "sum[fore,bold]"
```

property FormatAppearance.Gradient as Color

Renders the text with a gradient color.

Type	Description
Color	A Color expression to define the second gradient color to end with.

By default, the Gradient property is 0, which indicates no effect. The GradientMode property specifies the way the gradient text is shown.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.GradientMode as Long

Indicates the gradient mode to be applied on text.

Type	Description
Long	A Long expression that indicates the way the gradient text is shown. The valid value is between 0 and 4.

By default, the GradientMode property is 1. The GradientMode property specifies the way the gradient text is shown.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.Italic as Boolean

Renders as italic text.

Type	Description
Boolean	A Boolean expression that specifies whether the text is rendered in italics.

By default, the Italic property is False. Use the Italic property on True, to show the objects in *italics*.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.Key as String

Indicates the key of the FormatAppearance object.

Type	Description
String	A String expression that specifies the unique key of the FormatAppearance object. The Key of the FormatAppearance should include only alpha-numeric characters, any other character will be ignored.

The Key property indicates the key of the FormatAppearance. *Use the Key of the FormatAppearance in [] to apply the specified object to any column or row. For instance, the PivotRows property on "0[bold,italic]" to group by the first column and shows the column's content in bold and italic. The same rule is applied to PivotColumns or PivotTotals property.*

By default, the FormatAppearances collection contains the following keys:

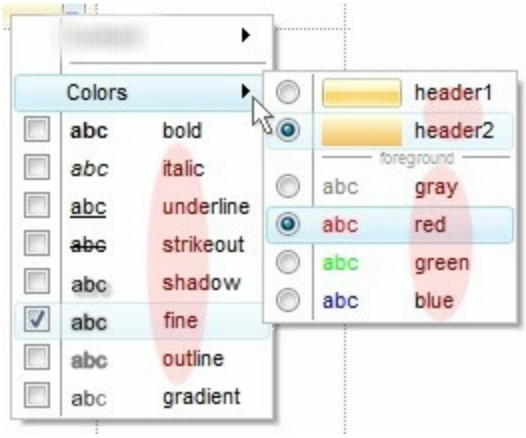
- **"bold",**
- **"italic",**
- **"strikeout"**
- **"underline"**

property FormatAppearance.Name as String

Specifies the name of the FormatAppearance object to be displayed on the context menu.

Type	Description
String	A String expression that defines the HTML caption to be displayed on the control's context menu.

The Name property indicates the name to be displayed on the control's context menu as shown bellow. The Key property specifies the key of the FormatAppearance object. The ToolTip property defines the FormatAppearance's tooltip.



The Name property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text

with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the

red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

property FormatAppearance.Outline as Color

Renders the text outlined with specified color.

Type	Description
Color	A Color expression that defines the color to show the outline text.

By default, the Outline property is 0, which indicates no effect. The OutlineSize property specifies the size of the outlined text/line.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.OutlineSize as Long

Indicates the size of the outline to be applied on text.

Type	Description
Long	A Long expression that defines the size of the outline text.

By default, the OutlineSize property is 1. If 0, the Outline has no effect. The OutlineSize property specifies the size of the outlined text/line.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.Shadow as Color

Renders the text with a shadow of specified color.

Type	Description
Color	A Color expression that defines the color of the shadow to be shown around the text

By default, the Shadow property is 0. If 0, this property has no effect. The ShadowOffset property defines the offset to show the text's shadow. The ShadowSize property define the size of the shadow to be shown.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.ShadowOffset as Long

Indicates the offset of the shadow to be applied on text.

Type	Description
Long	A Long expression that defines the offset where the text's shadow is shown.

By default, the ShadowOffset property is 2. If 0, this property shows the fine effect. The ShadowOffset property defines the offset to show the text's shadow. The ShadowColor property defines the color to show the text's shadow. The ShadowSize property define the size of the shadow to be shown.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.ShadowSize as Long

Indicates the size of the shadow to be applied on text.

Type	Description
Long	A Long expression that defines the size of the text's shadow.

By default, the ShadowSize property is 4. If 0, this property shows no shadow effect. The ShadowOffset property defines the offset to show the text's shadow. The ShadowColor property defines the color to show the text's shadow. The ShadowSize property define the size of the shadow to be shown.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.StrikeOut as Boolean

Specifies that the text should appear as strikeout.

Type	Description
Boolean	A Boolean expression that specifies whether the text is strikeout.

By default, the StrikeOut property is False. Use the StrikeOut property on True, to show the objects as ~~strikeout~~.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

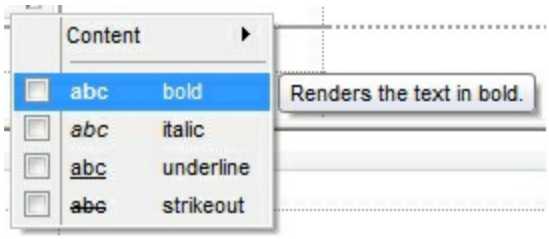
- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatAppearance.ToolTip as String

Specifies the tooltip of the FormatAppearance object to be displayed when the cursor hovers the object.

Type	Description
String	A String expression that defines the HTML caption to be shown when the cursor hovers the FormatAppearance object on the control's context menu.

The ToolTip property defines the FormatAppearance's tooltip. The Name property indicates the name to be displayed on the control's context menu. The Key property specifies the key of the FormatAppearance object.



The Name property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The

HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

property FormatAppearance.Underline as Boolean

Underlines the text.

Type	Description
Boolean	A Boolean expression that specifies whether the text is underlined.

By default, the Underline property is False. Use the Underline property on True, to show the objects underlined.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

FormatAppearances object

The FormatAppearances collection can be accessed through the control's FormatAppearances property. The FormatAppearance object changes the visual appearance of your data as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

By default, the control's context menu displays the following FormatAppearance objects:

By default, the FormatAppearances collection contains the following keys:

- **"bold",**
- **"italic",**
- **"strikeout"**
- **"underline"**

The FormatAppearances collection supports the following properties and methods:

Name	Description
Add	Adds a FormatAppearance object and returns a reference to the newly created object.
Clear	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
Item	Returns a specific FormatAppearance object giving its key.
Remove	Removes a specific member from the collection.

method FormatAppearances.Add (Key as String, [Name as Variant])

Adds a FormatAppearance object and returns a reference to the newly created object.

Type	Description
Key as String	A String expression that specifies the unique key of the FormatAppearance object. The Key of the FormatAppearance should include only alpha-numeric characters, any other character will be ignored.
Name as Variant	A String expression that indicates the HTML caption to be displayed on the control's context menu.

Return	Description
FormatAppearance	A FormatAppearance object being created.

By default, the FormatAppearances collection contains the following keys: "bold", "italic", "underline" and "strikeout". Use the Clear method to remove all FormatAppearance objects from the FormatAppearances collection. Use the Remove method to remove a FormatAppearance object giving its key. Use the Item property to access a FormatAppearance object giving its key. The Key property indicates the key of the FormatAppearance object.

The FormatAppearance object changes the visual appearance of your data as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

Use the Key of the FormatAppearance in [] to apply the specified object to any column or row. For instance, the PivotRows property on "0[bold,italic]" to group by the first column and shows the column's content in bold and italic. The same rule is applied to PivotColumns or PivotTotals property.

method FormatAppearances.Clear ()

Removes all objects in a collection.

Type	Description

The Clear method removes all elements in the FormatAppearances collection. Excludes the exPivotBarAllowFormatAppearance flag from the PivotBarVisible property, and so no FormatAppearance objects are displayed on the control's context menu. The Remove method removes a FormatAppearance object giving its key. Use the Add method to add a new FormatAppearance object. You can use the **for each** statement to enumerate all objects in the FormatAppearances collection.

property FormatAppearances.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A Long expression that indicates the number of FormatAppearance objects in the FormatAppearances collection.

The Count property gets the number of FormatAppearance objects in the FormatAppearances collection. The Clear method removes all elements in the FormatAppearances collection. The Item property accesses a FormatAppearance object based on its key. Use the Add method to add a new FormatAppearance object. You can use the **for each** statement to enumerate all objects in the FormatAppearances collection.

The following samples show how to change the "bold" caption being displayed in the control's context menu.

VBA (MS Access, Excell...)

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

VB6

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

VB.NET

```
With Expivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

VB.NET for /COM

```
With AxPivot1
```

```
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->GetFormatAppearances()->GetItem("bold")->PutName(L"Ingrosat");
```

C++ Builder

```
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->FormatAppearances->get_Item(TVariant("bold"))->Name = L"Ingrosat";
```

C#

```
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.FormatAppearances["bold"].Name = "Ingrosat";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
    Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
```

```
   Pivot1.FormatAppearances.Item("bold").Name = "Ingrosat";
</SCRIPT>
```

C# for /COM

```
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.FormatAppearances["bold"].Name = "Ingrosat";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatAppearance;
   anytype var_FormatAppearance;
   ;

   super();

   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   var_FormatAppearance =
COM::createFromObject(expivot1.FormatAppearances()).Item("bold");
com_FormatAppearance = var_FormatAppearance;
   com_FormatAppearance.Name("Ingrosat");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   FormatAppearances.Item['bold'].Name := 'Ingrosat';
end
```

Delphi (standard)

```
with Pivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
```

```
   FormatAppearances.Item['bold'].Name := 'Ingrosat';
end
```

VFP

```
with thisform.Pivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatAppearances.Item("bold").Name = "Ingrosat"
endwith
```

dBASE Plus

```
local oPivot,var_FormatAppearance

oPivot = form.Activex1.nativeObject
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
// oPivot.FormatAppearances.Item("bold").Name = "Ingrosat"
var_FormatAppearance = oPivot.FormatAppearances.Item("bold")
with (oPivot)
   TemplateDef = [Dim var_FormatAppearance]
   TemplateDef = var_FormatAppearance
   Template = [var_FormatAppearance.Name = "Ingrosat"]
endwith
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
' oPivot.FormatAppearances.Item("bold").Name = "Ingrosat"
var_FormatAppearance = oPivot.FormatAppearances.Item("bold")
oPivot.TemplateDef = "Dim var_FormatAppearance"
oPivot.TemplateDef = var_FormatAppearance
oPivot.Template = "var_FormatAppearance.Name = \"Ingrosat\""
```

Visual Objects

```
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:FormatAppearances:[Item,"bold"]:Name := "Ingrosat"
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.FormatAppearances.Item("bold").Name = "Ingrosat"
```

property FormatAppearances.Item (Key as Variant) as FormatAppearance

Returns a specific FormatAppearance object giving its key.

Type	Description
Key as Variant	A String expression that specifies the key of the object to be retrieved.
FormatAppearance	A FormatAppearance object being requested.

The Item property accesses a FormatAppearance object based on its key. Use the Add method to add a new FormatAppearance object. The Count property gets the number of FormatAppearance objects in the FormatAppearances collection. You can use the **for each** statement to enumerate all objects in the FormatAppearances collection.

The following samples show how to change the "bold" caption being displayed in the control's context menu.

VBA (MS Access, Excell...)

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

VB6

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

VB.NET

```
With Expivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

VB.NET for /COM

```
With AxPivot1
```

```
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .FormatAppearances.Item("bold").Name = "Ingrosat"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->GetFormatAppearances()->GetItem("bold")->PutName(L"Ingrosat");
```

C++ Builder

```
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->FormatAppearances->get_Item(TVariant("bold"))->Name = L"Ingrosat";
```

C#

```
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.FormatAppearances["bold"].Name = "Ingrosat";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
    Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
```

```
   Pivot1.FormatAppearances.Item("bold").Name = "Ingrosat";
</SCRIPT>
```

C# for /COM

```
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.FormatAppearances["bold"].Name = "Ingrosat";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatAppearance;
   anytype var_FormatAppearance;
   ;

   super();

   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   var_FormatAppearance =
COM::createFromObject(expivot1.FormatAppearances()).Item("bold");
com_FormatAppearance = var_FormatAppearance;
   com_FormatAppearance.Name("Ingrosat");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   FormatAppearances.Item['bold'].Name := 'Ingrosat';
end
```

Delphi (standard)

```
with Pivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
```

```
   FormatAppearances.Item['bold'].Name := 'Ingrosat';
end
```

VFP

```
with thisform.Pivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatAppearances.Item("bold").Name = "Ingrosat"
endwith
```

dBASE Plus

```
local oPivot,var_FormatAppearance

oPivot = form.Activex1.nativeObject
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
// oPivot.FormatAppearances.Item("bold").Name = "Ingrosat"
var_FormatAppearance = oPivot.FormatAppearances.Item("bold")
with (oPivot)
   TemplateDef = [Dim var_FormatAppearance]
   TemplateDef = var_FormatAppearance
   Template = [var_FormatAppearance.Name = "Ingrosat"]
endwith
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
' oPivot.FormatAppearances.Item("bold").Name = "Ingrosat"
var_FormatAppearance = oPivot.FormatAppearances.Item("bold")
oPivot.TemplateDef = "Dim var_FormatAppearance"
oPivot.TemplateDef = var_FormatAppearance
oPivot.Template = "var_FormatAppearance.Name = \"Ingrosat\""
```

Visual Objects

```
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:FormatAppearances:[Item,"bold"]:Name := "Ingrosat"
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.FormatAppearances.Item("bold").Name = "Ingrosat"
```

method FormatAppearances.Remove (Key as Variant)

Removes a specific member from the collection.

Type	Description
Key as Variant	A String expression that indicates the Key of the FormatAppearance object to be removed.

The Remove method removes a FormatAppearance object giving its key. The Clear method removes all elements in the FormatAppearances collection. Excludes the exPivotBarAllowFormatAppearance flag from the PivotBarVisible property, and so no FormatAppearance objects are displayed on the control's context menu. Use the Add method to add a new FormatAppearance object. You can use the **for each** statement to enumerate all objects in the FormatAppearances collection.

FormatConditionalAppearance object

The FormatConditionalAppearance objects can be accessed through the control's FormatConditionalAppearances property. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value. The FormatConditionalAppearance object changes the visual appearance of your data as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

The following screen shot shows the control's context menu with the FormatConditionalAppearance objects:

The FormatConditionalAppearance object supports the following properties and methods:

Name	Description
BackColor	Specifies the element's background color.
Bold	Renders as bold text.
ContextEditExpression	Indicates whether the item of the current conditional-format object in the control's content menu displays/edits the conditional expression, at runtime.
Expression	Specifies the conditional expression that determines whether the current format is applied on the pivot's data.
Font	Retrieves or sets the text's font.
FontSize	Indicates the size of the font to display the text.
ForeColor	Specifies the element's foreground color.
Gradient	Renders the text with a gradient color.
GradientMode	Indicates the gradient mode to be applied on text.
Italic	Renders as italic text.
	Indicates the key of the FormatConditionalAppearance

Key	object.
Name	Specifies the name of the FormatConditionalAppearance object to be displayed on the context menu.
Outline	Renders the text outlined with specified color.
OutlineSize	Indicates the size of the outline to be applied on text.
Shadow	Renders the text with a shadow of specified color.
ShadowOffset	Indicates the offset of the shadow to be applied on text.
ShadowSize	Indicates the size of the shadow to be applied on text.
StrikeOut	Specifies that the text should appear as strikeout.
ToolTip	Specifies the tooltip of the FormatConditionalAppearance object to be displayed when the cursor hovers the object.
Underline	Underlines the text.

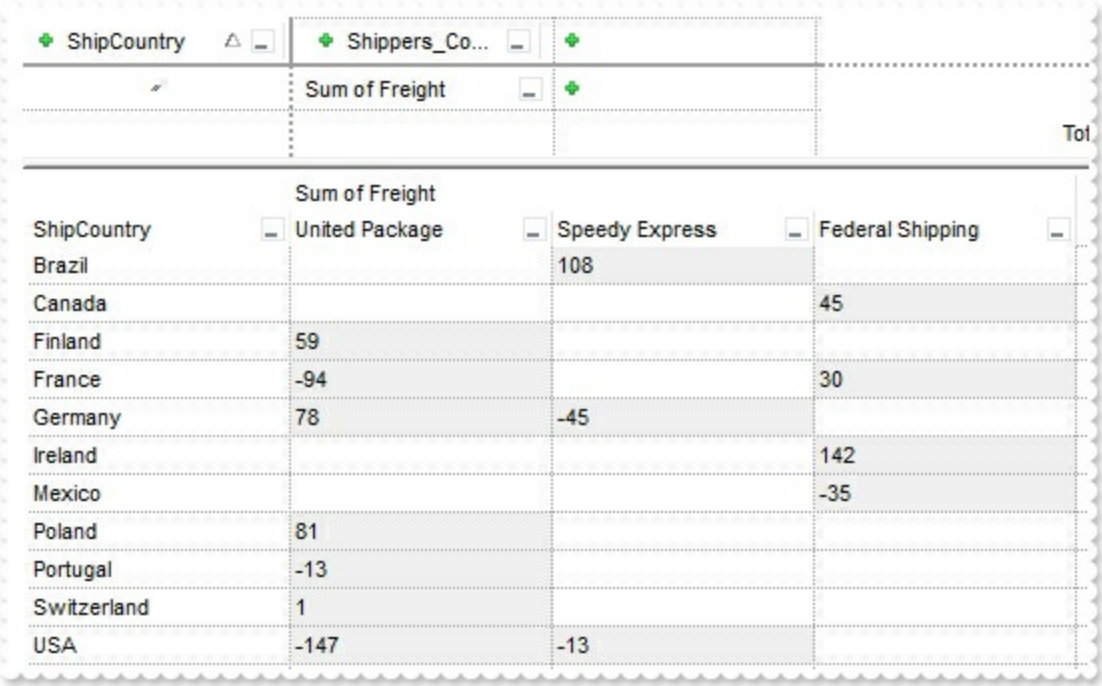
property FormatConditionalAppearance.BackColor as Color

Specifies the element's background color.

Type	Description
Color	A color expression that defines the background color to be applied. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The BackColor property defines the background color to be applied. This property supports solid colors or EBN colors. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part. The ForeColor property defines the foreground color to be applied on the objects (columns/rows). The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following samples show how you can mark not-empty values in the chart:



VBA (MS Access, Excell...)

```
With Pivot1
   .BeginUpdate
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   With .FormatConditionalAppearances.Add("nempty","not empty","")
      .Expression = "len(value) != 0"
      .BackColor = RGB(240,240,240)
   End With
   .PivotColumns = "sum(5)[nempty]/12"
   .EndUpdate
End With
```

VB6

```
With Pivot1
   .BeginUpdate
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   With .FormatConditionalAppearances.Add("nempty","not empty","")
      .Expression = "len(value) != 0"
      .BackColor = RGB(240,240,240)
   End With
   .PivotColumns = "sum(5)[nempty]/12"
   .EndUpdate
End With
```

VB.NET

```
With Expivot1
   .BeginUpdate()
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
   With .FormatConditionalAppearances.Add("nempty","not empty","")
      .Expression = "len(value) != 0"
      .BackColor = Color.FromArgb(240,240,240)
   End With
   .PivotColumns = "sum(5)[nempty]/12"
   .EndUpdate()
```

```
End With
```

VB.NET for /COM

```
With AxPivot1
   .BeginUpdate()
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
   With .FormatConditionalAppearances.Add("nempty","not empty","")
      .Expression = "len(value) != 0"
      .BackColor = RGB(240,240,240)
   End With
   .PivotColumns = "sum(5)[nempty]/12"
   .EndUpdate()
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

   #import <ExPivot.dll>
   using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->BeginUpdate();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->PutPivotRows(L"0");
EXPIVOTLib::IFormatConditionalAppearancePtr var_FormatConditionalAppearance =
spPivot1->GetFormatConditionalAppearances()->Add(L"nempty","not empty","");
   var_FormatConditionalAppearance->PutExpression(L"len(value) != 0");
   var_FormatConditionalAppearance->PutBackColor(RGB(240,240,240));
spPivot1->PutPivotColumns(L"sum(5)[nempty]/12");
spPivot1->EndUpdate();
```

C++ Builder

```
Pivot1->BeginUpdate();
Pivot1->Import(TVariant("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->PivotRows = L"0";
Expivotlib_tlb::IFormatConditionalAppearancePtr var_FormatConditionalAppearance = Pivot1->FormatConditionalAppearances->Add(L"nempty",TVariant("not empty"),TVariant(""));
   var_FormatConditionalAppearance->Expression = L"len(value) != 0";
   var_FormatConditionalAppearance->BackColor = RGB(240,240,240);
Pivot1->PivotColumns = L"sum(5)[nempty]/12";
Pivot1->EndUpdate();
```

C#

```
expivot1.BeginUpdate();
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.PivotRows = "0";
exontrol.EXPIVOTLib.FormatConditionalAppearance var_FormatConditionalAppearance = expivot1.FormatConditionalAppearances.Add("nempty","not empty","");
   var_FormatConditionalAppearance.Expression = "len(value) != 0";
   var_FormatConditionalAppearance.BackColor = Color.FromArgb(240,240,240);
expivot1.PivotColumns = "sum(5)[nempty]/12";
expivot1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
```

```
   Pivot1.BeginUpdate();
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   Pivot1.PivotRows = "0";
   var var_FormatConditionalAppearance =
Pivot1.FormatConditionalAppearances.Add("nempty","not empty","");
      var_FormatConditionalAppearance.Expression = "len(value) != 0";
      var_FormatConditionalAppearance.BackColor = 15790320;
   Pivot1.PivotColumns = "sum(5)[nempty]/12";
   Pivot1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686"
id="Pivot1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With Pivot1
      .BeginUpdate
      .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
      .PivotRows = "0"
      With .FormatConditionalAppearances.Add("nempty","not empty","")
         .Expression = "len(value) != 0"
         .BackColor = RGB(240,240,240)
      End With
      .PivotColumns = "sum(5)[nempty]/12"
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axPivot1.BeginUpdate();
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.PivotRows = "0";
EXPIVOTLib.FormatConditionalAppearance var_FormatConditionalAppearance =
axPivot1.FormatConditionalAppearances.Add("nempty","not empty","");
   var_FormatConditionalAppearance.Expression = "len(value) != 0";
   var_FormatConditionalAppearance.BackColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(240,240,240));
axPivot1.PivotColumns = "sum(5)[nempty]/12";
axPivot1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatConditionalAppearance;
   anytype var_FormatConditionalAppearance;
   ;

   super();

   expivot1.BeginUpdate();
   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   expivot1.PivotRows("0");
   var_FormatConditionalAppearance =
COM::createFromObject(expivot1.FormatConditionalAppearances()).Add("nempty","no
 empty",""); com_FormatConditionalAppearance =
var_FormatConditionalAppearance;
      com_FormatConditionalAppearance.Expression("len(value) != 0");
      com_FormatConditionalAppearance.BackColor(WinApi::RGB2int(240,240,240));
   expivot1.PivotColumns("sum(5)[nempty]/12");
   expivot1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   BeginUpdate();
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   PivotRows := '0';
   with FormatConditionalAppearances.Add('nempty','not empty','') do
   begin
      Expression := 'len(value) != 0';
      BackColor := $f0f0f0;
   end;
   PivotColumns := 'sum(5)[nempty]/12';
   EndUpdate();
end
```

Delphi (standard)

```
with Pivot1 do
begin
   BeginUpdate();
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
   PivotRows := '0';
   with FormatConditionalAppearances.Add('nempty','not empty','') do
   begin
      Expression := 'len(value) != 0';
      BackColor := $f0f0f0;
   end;
   PivotColumns := 'sum(5)[nempty]/12';
   EndUpdate();
end
```

VFP

```
with thisform.Pivot1
   .BeginUpdate
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .PivotRows = "0"
   with .FormatConditionalAppearances.Add("nempty","not empty","")
      .Expression = "len(value) != 0"
```

```
      .BackColor = RGB(240,240,240)
   endwith
   .PivotColumns = "sum(5)[nempty]/12"
   .EndUpdate
endwith
```

dBASE Plus

```
local oPivot,var_FormatConditionalAppearance

oPivot = form.EXPIVOTACTIVEXCONTROL1.nativeObject
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("nempty","not empty","")
   var_FormatConditionalAppearance.Expression = "len(value) != 0"
   var_FormatConditionalAppearance.BackColor = 0xf0f0f0
oPivot.PivotColumns = "sum(5)[nempty]/12"
oPivot.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatConditionalAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("nempty","not empty","")
   var_FormatConditionalAppearance.Expression = "len(value) != 0"
   var_FormatConditionalAppearance.BackColor = 15790320
oPivot.PivotColumns = "sum(5)[nempty]/12"
oPivot.EndUpdate()
```

Visual Objects

```
local var_FormatConditionalAppearance as IFormatConditionalAppearance

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:PivotRows := "0"
var_FormatConditionalAppearance :=
oDCOCX_Exontrol1:FormatConditionalAppearances:Add("nempty","not empty","")
   var_FormatConditionalAppearance:Expression := "len(value) != 0"
   var_FormatConditionalAppearance:BackColor := RGB(240,240,240)
oDCOCX_Exontrol1:PivotColumns := "sum(5)[nempty]/12"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oPivot,var_FormatConditionalAppearance

oPivot = ole_1.Object
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("nempty","not empty","")
   var_FormatConditionalAppearance.Expression = "len(value) != 0"
   var_FormatConditionalAppearance.BackColor = RGB(240,240,240)
oPivot.PivotColumns = "sum(5)[nempty]/12"
oPivot.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Get ComImport "C:\Program Files\Exontrol\ExPivot\Sample\data.txt" Nothing to
Nothing
```

```
   Set ComPivotRows to "0"
   Variant voFormatConditionalAppearances
   Get ComFormatConditionalAppearances to voFormatConditionalAppearances
   Handle hoFormatConditionalAppearances
   Get Create (RefClass(cComFormatConditionalAppearances)) to
hoFormatConditionalAppearances
   Set pvComObject of hoFormatConditionalAppearances to
voFormatConditionalAppearances
      Variant voFormatConditionalAppearance
      Get ComAdd of hoFormatConditionalAppearances "nempty" "not empty" "" to
voFormatConditionalAppearance
      Handle hoFormatConditionalAppearance
      Get Create (RefClass(cComFormatConditionalAppearance)) to
hoFormatConditionalAppearance
      Set pvComObject of hoFormatConditionalAppearance to
voFormatConditionalAppearance
         Set ComExpression of hoFormatConditionalAppearance to "len(value) != 0"
         Set ComBackColor of hoFormatConditionalAppearance to (RGB(240,240,240))
      Send Destroy to hoFormatConditionalAppearance
   Send Destroy to hoFormatConditionalAppearances
   Set ComPivotColumns to "sum(5)[nempty]/12"
   Send ComEndUpdate
End_Procedure
```

XBase++

```
#include "AppEvent.ch"
#include "ActiveX.ch"

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oFormatConditionalAppearance
   LOCAL oPivot

   oForm := XbpDialog():new( AppDesktop() )
   oForm:drawingArea:clipChildren := .T.
```

```
   oForm:create( ,,{100,100}, {640,480},, .F. )
   oForm:close  := {|| PostAppEvent( xbeP_Quit )}

   oPivot := XbpActiveXControl():new( oForm:drawingArea )
   oPivot:CLSID  := "Exontrol.Pivot.1" /*{5C9DF3D3-81B1-42C4-BED6-
658F17748686}*/
   oPivot:create(,, {10,60},{610,370} )

      oPivot:BeginUpdate()
      oPivot:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
      oPivot:PivotRows := "0"
      oFormatConditionalAppearance :=
oPivot:FormatConditionalAppearances():Add("nempty","not empty","")
         oFormatConditionalAppearance:Expression := "len(value) != 0"

oFormatConditionalAppearance:SetProperty("BackColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 240,240,240 } )  , .F. ))
      oPivot:PivotColumns := "sum(5)[nempty]/12"
      oPivot:EndUpdate()

   oForm:Show()
   DO WHILE nEvent != xbeP_Quit
      nEvent := AppEvent( @mp1, @mp2, @oXbp )
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO
RETURN
```

property FormatConditionalAppearance.Bold as Boolean

Renders as bold text.

Type	Description
Boolean	A Boolean expression that specifies whether the text is rendered in bold.

By default, the Bold property is False. Use the Bold property on True, to show the objects in **bold**. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

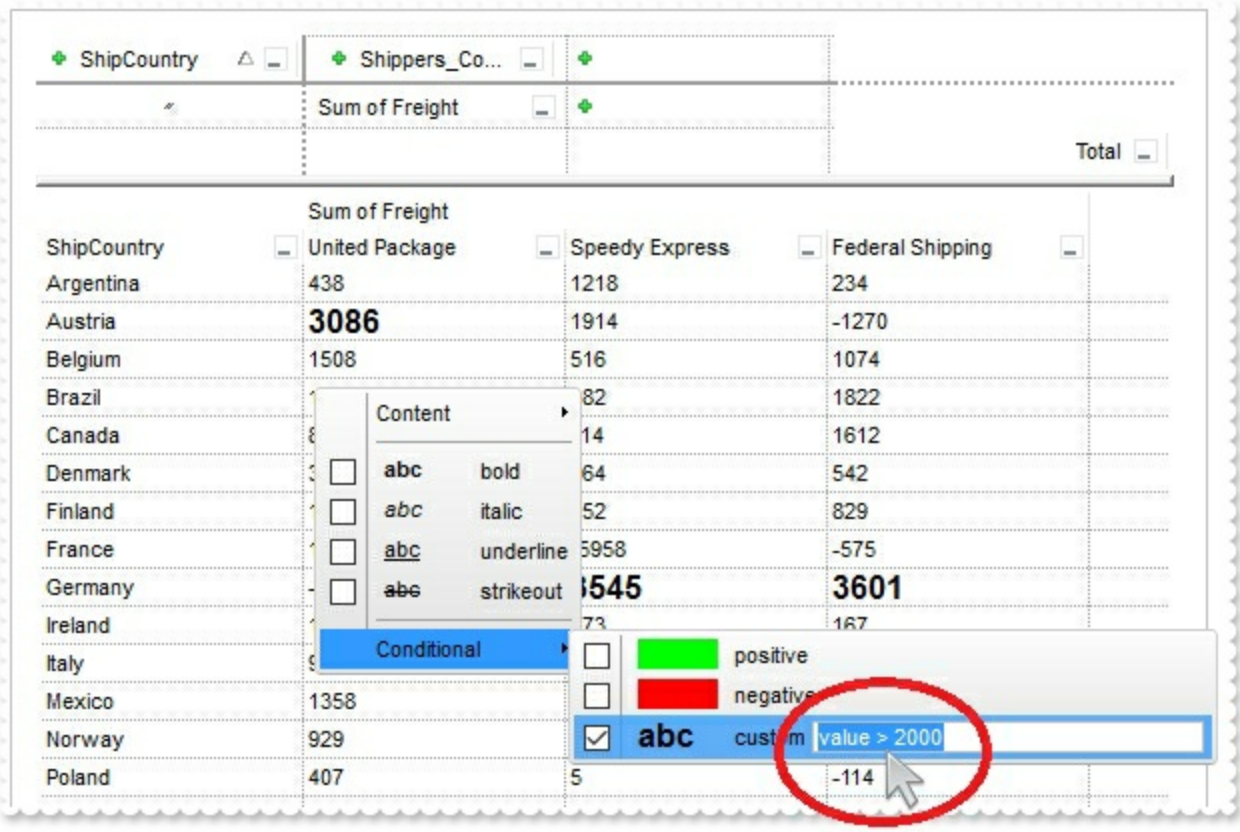
property FormatConditionalAppearance.ContextEditExpression as Boolean

Indicates whether the item of the current conditional-format object in the control's content menu displays/edits the conditional expression, at runtime.

Type	Description
Boolean	A Boolean expression that indicates whether the item of the current conditional-format object in the control's content menu displays/edits the conditional expression, at runtime.

By default, the ContextEditExpression property is False, so the format's expression is not editable at runtime. Use the ContextEditExpression property on True, to allow user to edit the format's Expression at runtime. The **value** keyword in the Expression property indicates the value/result to be evaluated. For instance, "value > 2000" indicates all values greater than 2000. While typing the FormatConditionalAppearance's Expression the item gets checked, and it shows as disabled if the Expression is not valid (empty or syntactically incorrect). The FormatConditionalAppearance format is not applied to any value, if not checked, or the expression is not valid (empty or syntactically incorrect)

The following screen show shows the "custom" FormatConditionalAppearance object, that allows editing the Expression property:



The following samples shows how you can edit the conditional-expression at runtime:

VBA (MS Access, Excell...)

```
With Pivot1
   .BeginUpdate
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   With .FormatConditionalAppearances.Add("custom","custom","")
      .Bold = True
      .FontSize = 12
      .Expression = "value > 2000"
      .ContextEditExpression = True
   End With
   .PivotRows = "0"
   .PivotColumns = "sum(5)[custom]/12"
   .EndUpdate
End With
```

VB6

```
With Pivot1
   .BeginUpdate
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   With .FormatConditionalAppearances.Add("custom","custom","")
      .Bold = True
      .FontSize = 12
      .Expression = "value > 2000"
      .ContextEditExpression = True
   End With
   .PivotRows = "0"
   .PivotColumns = "sum(5)[custom]/12"
   .EndUpdate
End With
```

VB.NET

```
With Expivot1
   .BeginUpdate()
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   With .FormatConditionalAppearances.Add("custom","custom","")
```

```
        .Bold = True
        .FontSize = 12
        .Expression = "value > 2000"
        .ContextEditExpression = True
    End With
    .PivotRows = "0"
    .PivotColumns = "sum(5)[custom]/12"
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxPivot1
    .BeginUpdate()
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    With .FormatConditionalAppearances.Add("custom","custom","")
        .Bold = True
        .FontSize = 12
        .Expression = "value > 2000"
        .ContextEditExpression = True
    End With
    .PivotRows = "0"
    .PivotColumns = "sum(5)[custom]/12"
    .EndUpdate()
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->BeginUpdate();
spPivot1->Import("C:\\Program
```

```
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
EXPIVOTLib::IFormatConditionalAppearancePtr var_FormatConditionalAppearance =
spPivot1->GetFormatConditionalAppearances()->Add(L"custom","custom","");
   var_FormatConditionalAppearance->PutBold(VARIANT_TRUE);
   var_FormatConditionalAppearance->PutFontSize(12);
   var_FormatConditionalAppearance->PutExpression(L"value > 2000");
   var_FormatConditionalAppearance->PutContextEditExpression(VARIANT_TRUE);
spPivot1->PutPivotRows(L"0");
spPivot1->PutPivotColumns(L"sum(5)[custom]/12");
spPivot1->EndUpdate();
```

C++ Builder

```
Pivot1->BeginUpdate();
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Expivotlib_tlb::IFormatConditionalAppearancePtr var_FormatConditionalAppearance
= Pivot1->FormatConditionalAppearances-
>Add(L"custom",TVariant("custom"),TVariant(""));
   var_FormatConditionalAppearance->Bold = true;
   var_FormatConditionalAppearance->FontSize = 12;
   var_FormatConditionalAppearance->Expression = L"value > 2000";
   var_FormatConditionalAppearance->ContextEditExpression = true;
Pivot1->PivotRows = L"0";
Pivot1->PivotColumns = L"sum(5)[custom]/12";
Pivot1->EndUpdate();
```

C#

```
expivot1.BeginUpdate();
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
exontrol.EXPIVOTLib.FormatConditionalAppearance
var_FormatConditionalAppearance =
expivot1.FormatConditionalAppearances.Add("custom","custom","");
   var_FormatConditionalAppearance.Bold = true;
   var_FormatConditionalAppearance.FontSize = 12;
```

```
   var_FormatConditionalAppearance.Expression = "value > 2000";
   var_FormatConditionalAppearance.ContextEditExpression = true;
expivot1.PivotRows = "0";
expivot1.PivotColumns = "sum(5)[custom]/12";
expivot1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686"
id="Pivot1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   Pivot1.BeginUpdate();
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   var var_FormatConditionalAppearance =
Pivot1.FormatConditionalAppearances.Add("custom","custom","");
      var_FormatConditionalAppearance.Bold = true;
      var_FormatConditionalAppearance.FontSize = 12;
      var_FormatConditionalAppearance.Expression = "value > 2000";
      var_FormatConditionalAppearance.ContextEditExpression = true;
   Pivot1.PivotRows = "0";
   Pivot1.PivotColumns = "sum(5)[custom]/12";
   Pivot1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686"
id="Pivot1"></OBJECT>
```

```
<SCRIPT LANGUAGE="VBScript">
Function Init()
   With Pivot1
      .BeginUpdate
      .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
      With .FormatConditionalAppearances.Add("custom","custom","")
         .Bold = True
         .FontSize = 12
         .Expression = "value > 2000"
         .ContextEditExpression = True
      End With
      .PivotRows = "0"
      .PivotColumns = "sum(5)[custom]/12"
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axPivot1.BeginUpdate();
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
EXPIVOTLib.FormatConditionalAppearance var_FormatConditionalAppearance =
axPivot1.FormatConditionalAppearances.Add("custom","custom","");
   var_FormatConditionalAppearance.Bold = true;
   var_FormatConditionalAppearance.FontSize = 12;
   var_FormatConditionalAppearance.Expression = "value > 2000";
   var_FormatConditionalAppearance.ContextEditExpression = true;
axPivot1.PivotRows = "0";
axPivot1.PivotColumns = "sum(5)[custom]/12";
axPivot1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
```

```
{
   COM com_FormatConditionalAppearance;
   anytype var_FormatConditionalAppearance;
   ;

   super();

   expivot1.BeginUpdate();
   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   var_FormatConditionalAppearance =
COM::createFromObject(expivot1.FormatConditionalAppearances()).Add("custom","cu
 com_FormatConditionalAppearance = var_FormatConditionalAppearance;
      com_FormatConditionalAppearance.Bold(true);
      com_FormatConditionalAppearance.FontSize(12);
      com_FormatConditionalAppearance.Expression("value > 2000");
      com_FormatConditionalAppearance.ContextEditExpression(true);
   expivot1.PivotRows("0");
   expivot1.PivotColumns("sum(5)[custom]/12");
   expivot1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   BeginUpdate();
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   with FormatConditionalAppearances.Add('custom','custom','') do
   begin
      Bold := True;
      FontSize := 12;
      Expression := 'value > 2000';
      ContextEditExpression := True;
   end;
   PivotRows := '0';
   PivotColumns := 'sum(5)[custom]/12';
   EndUpdate();
```

```
end
```

Delphi (standard)

```
with Pivot1 do
begin
   BeginUpdate();
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
   with FormatConditionalAppearances.Add('custom','custom','') do
   begin
      Bold := True;
      FontSize := 12;
      Expression := 'value > 2000';
      ContextEditExpression := True;
   end;
   PivotRows := '0';
   PivotColumns := 'sum(5)[custom]/12';
   EndUpdate();
end
```

VFP

```
with thisform.Pivot1
   .BeginUpdate
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   with .FormatConditionalAppearances.Add("custom","custom","")
      .Bold = .T.
      .FontSize = 12
      .Expression = "value > 2000"
      .ContextEditExpression = .T.
   endwith
   .PivotRows = "0"
   .PivotColumns = "sum(5)[custom]/12"
   .EndUpdate
endwith
```

dBASE Plus

```
local oPivot,var_FormatConditionalAppearance

oPivot = form.EXPIVOTACTIVEXCONTROL1.nativeObject
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("custom","custom","")
   var_FormatConditionalAppearance.Bold = true
   var_FormatConditionalAppearance.FontSize = 12
   var_FormatConditionalAppearance.Expression = "value > 2000"
   var_FormatConditionalAppearance.ContextEditExpression = true
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)[custom]/12"
oPivot.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatConditionalAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("custom","custom","")
   var_FormatConditionalAppearance.Bold = .t.
   var_FormatConditionalAppearance.FontSize = 12
   var_FormatConditionalAppearance.Expression = "value > 2000"
   var_FormatConditionalAppearance.ContextEditExpression = .t.
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)[custom]/12"
oPivot.EndUpdate()
```

Visual Objects

```
local var_FormatConditionalAppearance as IFormatConditionalAppearance
```

```
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
var_FormatConditionalAppearance :=
oDCOCX_Exontrol1:FormatConditionalAppearances:Add("custom","custom","")
   var_FormatConditionalAppearance:Bold := true
   var_FormatConditionalAppearance:FontSize := 12
   var_FormatConditionalAppearance:Expression := "value > 2000"
   var_FormatConditionalAppearance:ContextEditExpression := true
oDCOCX_Exontrol1:PivotRows := "0"
oDCOCX_Exontrol1:PivotColumns := "sum(5)[custom]/12"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oPivot,var_FormatConditionalAppearance

oPivot = ole_1.Object
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("custom","custom","")
   var_FormatConditionalAppearance.Bold = true
   var_FormatConditionalAppearance.FontSize = 12
   var_FormatConditionalAppearance.Expression = "value > 2000"
   var_FormatConditionalAppearance.ContextEditExpression = true
oPivot.PivotRows = "0"
oPivot.PivotColumns = "sum(5)[custom]/12"
oPivot.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Get ComImport "C:\Program Files\Exontrol\ExPivot\Sample\data.txt" Nothing to
```

```
Nothing
   Variant voFormatConditionalAppearances
   Get ComFormatConditionalAppearances to voFormatConditionalAppearances
   Handle hoFormatConditionalAppearances
   Get Create (RefClass(cComFormatConditionalAppearances)) to
hoFormatConditionalAppearances
   Set pvComObject of hoFormatConditionalAppearances to
voFormatConditionalAppearances
      Variant voFormatConditionalAppearance
      Get ComAdd of hoFormatConditionalAppearances "custom" "custom" "" to
voFormatConditionalAppearance
      Handle hoFormatConditionalAppearance
      Get Create (RefClass(cComFormatConditionalAppearance)) to
hoFormatConditionalAppearance
      Set pvComObject of hoFormatConditionalAppearance to
voFormatConditionalAppearance
         Set ComBold of hoFormatConditionalAppearance to True
         Set ComFontSize of hoFormatConditionalAppearance to 12
         Set ComExpression of hoFormatConditionalAppearance to "value > 2000"
         Set ComContextEditExpression of hoFormatConditionalAppearance to True
      Send Destroy to hoFormatConditionalAppearance
   Send Destroy to hoFormatConditionalAppearances
   Set ComPivotRows to "0"
   Set ComPivotColumns to "sum(5)[custom]/12"
   Send ComEndUpdate
End_Procedure
```

XBase++

```
#include "AppEvent.ch"
#include "ActiveX.ch"

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oFormatConditionalAppearance
   LOCAL oPivot
```

```
   oForm := XbpDialog():new( AppDesktop() )
   oForm:drawingArea:clipChildren := .T.
   oForm:create( ,,{100,100}, {640,480},, .F. )
   oForm:close  := {|| PostAppEvent( xbeP_Quit )}

   oPivot := XbpActiveXControl():new( oForm:drawingArea )
   oPivot:CLSID  := "Exontrol.Pivot.1" /*{5C9DF3D3-81B1-42C4-BED6-
658F17748686}*/
   oPivot:create(,, {10,60},{610,370} )

      oPivot:BeginUpdate()
      oPivot:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
      oFormatConditionalAppearance :=
oPivot:FormatConditionalAppearances():Add("custom","custom","")
         oFormatConditionalAppearance:Bold := .T.
         oFormatConditionalAppearance:FontSize := 12
         oFormatConditionalAppearance:Expression := "value > 2000"
         oFormatConditionalAppearance:ContextEditExpression := .T.
      oPivot:PivotRows := "0"
      oPivot:PivotColumns := "sum(5)[custom]/12"
      oPivot:EndUpdate()

   oForm:Show()
   DO WHILE nEvent != xbeP_Quit
      nEvent := AppEvent( @mp1, @mp2, @oXbp )
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO
RETURN
```

property FormatConditionalAppearance.Expression as String

Specifies the conditional expression that determines whether the current format is applied on the pivot's data.

Type	Description
String	A String expression that defines the conditional-expression that needs to match the value, so the current format is applied to the cell. For instance, "len(value) = 0" indicates any empty value.

The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's value. The Expression parameter of the Add method, is equivalent with the Expression property. When the current format is applied to a column, the Expression is evaluated (value keyword in the Expression indicates the value in the column), and if it is True, the current format is applied to the cell, else it is not. The ContextEditExpression property indicates whether the item of the current conditional-format object in the control's content menu displays/edits the conditional expression, at runtime. The FormatConditionalAppearance format is not applied to any value, if the expression is not valid (empty or syntactically incorrect).

For instance:

- *"len(value) != 0"*, indicates not-empty values
- *"len(value) = 0"*, indicates empty values
- *"value < 0"*, indicates negative value
- "*len(value) != 0 ? (value < 0) : 0*", indicates any negative value, by checking first it is it an not-empty cell/value

The **value** keyword in the Expression property indicates the value/result to be evaluated.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- ***/*** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
  - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

  - 0 - empty (not initialized)
  - 1 - null
  - 2 - short
  - 3 - long
  - 4 - float
  - 5 - double
  - 6 - currency
  - 7 - date
  - 8 - string
  - 9 - object
  - 10 - error
  - 11 - boolean
  - 12 - variant
  - 13 - any
  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long
  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not

greater than the value of its argument

- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
  - *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
  - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
    - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
    - 1 - Negative sign, number; for example, -1.1
    - 2 - Negative sign, space, number; for example, - 1.1
    - 3 - Number, negative sign; for example, 1.1-
    - 4 - Number, space, negative sign; for example, 1.1 -
  - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance "ABCABC" lfind "C" returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance "ABCABC" rfind "C" returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)

- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

property FormatConditionalAppearance.Font as IFontDisp

Retrieves or sets the text's font.

Type	Description
IFontDisp	A Font object that indicates the font to be applied by the FormatConditionalAppearance object.

By default, the Font property is nothing/empty, which indicates no effect. You can use the Font property to specify a different font to be applied on selected columns/rows. The FontSize property can be used to change just the size of the current font, which is applied to the current selection. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

property FormatConditionalAppearance.FontSize as Long

Indicates the size of the font to display the text.

Type	Description
Long	A Long expression that defines the size of the font

By default, the FontSize property is 0, which indicates no effect. The FontSize property can be used to change just the size of the current font, which is applied to the current selection. You can use the Font property to specify a different font to be applied on selected columns/rows. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

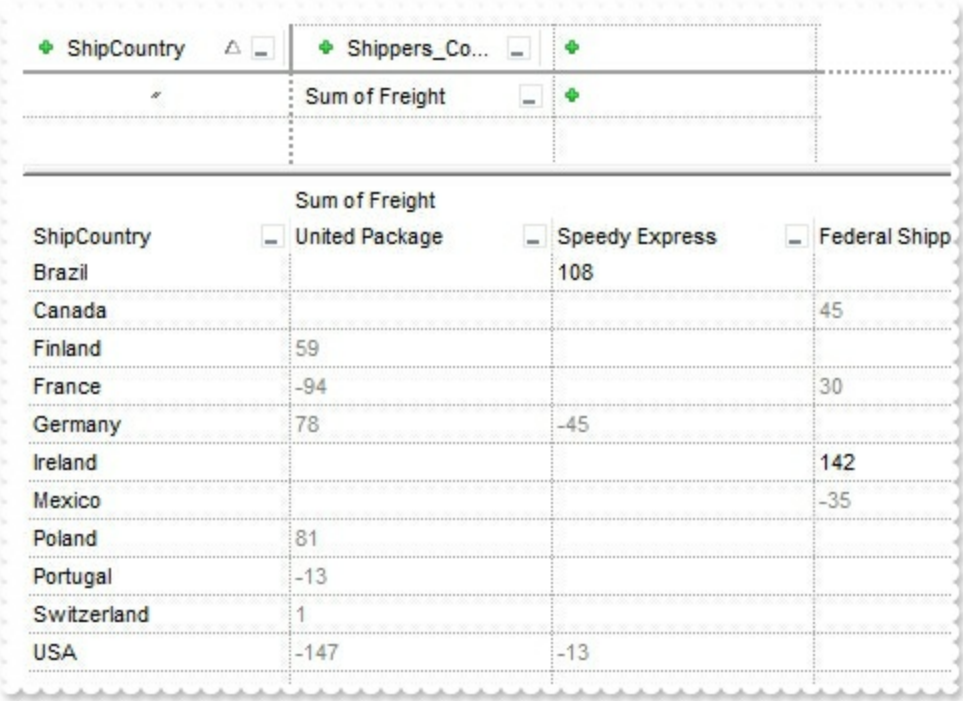
property FormatConditionalAppearance.ForeColor as Color

Specifies the element's foreground color.

Type	Description
Color	A Color expression that defines the color to be applied on the object's foreground.

The ForeColor property specifies the foreground color to be applied on the columns/rows. The BackColor property defines the background color to be applied on the objects (columns/rows). The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following sample shows how you can shows the value with a value less than 100, with a gray color:



VBA (MS Access, Excell...)

```
With Pivot1
   .BeginUpdate
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .PivotRows = "0"
   With .FormatConditionalAppearances.Add("lower","lower","")
      .Expression = "len(value) != 0 ? (value < 100) : 0"
      .ForeColor = RGB(128,128,128)
   End With
```

```
    .PivotColumns = "sum(5)[lower]/12"
    .EndUpdate
End With
```

VB6

```
With Pivot1
    .BeginUpdate
    .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
    .PivotRows = "0"
    With .FormatConditionalAppearances.Add("lower","lower","")
        .Expression = "len(value) != 0 ? (value < 100) : 0"
        .ForeColor = RGB(128,128,128)
    End With
    .PivotColumns = "sum(5)[lower]/12"
    .EndUpdate
End With
```

VB.NET

```
With Expivot1
    .BeginUpdate()
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .PivotRows = "0"
    With .FormatConditionalAppearances.Add("lower","lower","")
        .Expression = "len(value) != 0 ? (value < 100) : 0"
        .ForeColor = Color.FromArgb(128,128,128)
    End With
    .PivotColumns = "sum(5)[lower]/12"
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxPivot1
    .BeginUpdate()
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .PivotRows = "0"
```

```
    With .FormatConditionalAppearances.Add("lower","lower","")
        .Expression = "len(value) != 0 ? (value < 100) : 0"
        .ForeColor = RGB(128,128,128)
    End With
    .PivotColumns = "sum(5)[lower]/12"
    .EndUpdate()
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->BeginUpdate();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->PutPivotRows(L"0");
EXPIVOTLib::IFormatConditionalAppearancePtr var_FormatConditionalAppearance =
spPivot1->GetFormatConditionalAppearances()->Add(L"lower","lower","");
    var_FormatConditionalAppearance->PutExpression(L"len(value) != 0 ? (value <
100) : 0");
    var_FormatConditionalAppearance->PutForeColor(RGB(128,128,128));
spPivot1->PutPivotColumns(L"sum(5)[lower]/12");
spPivot1->EndUpdate();
```

C++ Builder

```
Pivot1->BeginUpdate();
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->PivotRows = L"0";
Expivotlib_tlb::IFormatConditionalAppearancePtr var_FormatConditionalAppearance
```

```
= Pivot1->FormatConditionalAppearances-
>Add(L"lower",TVariant("lower"),TVariant(""));
   var_FormatConditionalAppearance->Expression = L"len(value) != 0 ? (value <
100) : 0";
   var_FormatConditionalAppearance->ForeColor = RGB(128,128,128);
Pivot1->PivotColumns = L"sum(5)[lower]/12";
Pivot1->EndUpdate();
```

C#

```
expivot1.BeginUpdate();
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.PivotRows = "0";
exontrol.EXPIVOTLib.FormatConditionalAppearance
var_FormatConditionalAppearance =
expivot1.FormatConditionalAppearances.Add("lower","lower","");
   var_FormatConditionalAppearance.Expression = "len(value) != 0 ? (value < 100) :
0";
   var_FormatConditionalAppearance.ForeColor = Color.FromArgb(128,128,128);
expivot1.PivotColumns = "sum(5)[lower]/12";
expivot1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686"
id="Pivot1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   Pivot1.BeginUpdate();
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   Pivot1.PivotRows = "0";
   var var_FormatConditionalAppearance =
Pivot1.FormatConditionalAppearances.Add("lower","lower","");
```

```
        var_FormatConditionalAppearance.Expression = "len(value) != 0 ? (value < 100) : 0";
        var_FormatConditionalAppearance.ForeColor = 8421504;
    Pivot1.PivotColumns = "sum(5)[lower]/12";
    Pivot1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686"
id="Pivot1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With Pivot1
        .BeginUpdate
        .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
        .PivotRows = "0"
        With .FormatConditionalAppearances.Add("lower","lower","")
            .Expression = "len(value) != 0 ? (value < 100) : 0"
            .ForeColor = RGB(128,128,128)
        End With
        .PivotColumns = "sum(5)[lower]/12"
        .EndUpdate
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axPivot1.BeginUpdate();
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
```

```
axPivot1.PivotRows = "0";
EXPIVOTLib.FormatConditionalAppearance var_FormatConditionalAppearance =
axPivot1.FormatConditionalAppearances.Add("lower","lower","");
   var_FormatConditionalAppearance.Expression = "len(value) != 0 ? (value < 100) :
0";
   var_FormatConditionalAppearance.ForeColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(128,128,128));
axPivot1.PivotColumns = "sum(5)[lower]/12";
axPivot1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatConditionalAppearance;
   anytype var_FormatConditionalAppearance;
   ;

   super();

   expivot1.BeginUpdate();
   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   expivot1.PivotRows("0");
   var_FormatConditionalAppearance =
COM::createFromObject(expivot1.FormatConditionalAppearances()).Add("lower","lowe
 com_FormatConditionalAppearance = var_FormatConditionalAppearance;
      com_FormatConditionalAppearance.Expression("len(value) != 0 ? (value < 100) :
0");
      com_FormatConditionalAppearance.ForeColor(WinApi::RGB2int(128,128,128));
   expivot1.PivotColumns("sum(5)[lower]/12");
   expivot1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
```

```
  BeginUpdate();
  Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
  PivotRows := '0';
  with FormatConditionalAppearances.Add('lower','lower','') do
  begin
    Expression := 'len(value) != 0 ? (value < 100) : 0';
    ForeColor := $808080;
  end;
  PivotColumns := 'sum(5)[lower]/12';
  EndUpdate();
end
```

Delphi (standard)

```
with Pivot1 do
begin
  BeginUpdate();
  Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
  PivotRows := '0';
  with FormatConditionalAppearances.Add('lower','lower','') do
  begin
    Expression := 'len(value) != 0 ? (value < 100) : 0';
    ForeColor := $808080;
  end;
  PivotColumns := 'sum(5)[lower]/12';
  EndUpdate();
end
```

VFP

```
with thisform.Pivot1
  .BeginUpdate
  .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
  .PivotRows = "0"
  with .FormatConditionalAppearances.Add("lower","lower","")
    .Expression = "len(value) != 0 ? (value < 100) : 0"
    .ForeColor = RGB(128,128,128)
  endwith
```

```
   .PivotColumns = "sum(5)[lower]/12"
   .EndUpdate
endwith
```

dBASE Plus

```
local oPivot,var_FormatConditionalAppearance

oPivot = form.EXPIVOTACTIVEXCONTROL1.nativeObject
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("lower","lower","")
   var_FormatConditionalAppearance.Expression = "len(value) != 0 ? (value < 100) :
0"
   var_FormatConditionalAppearance.ForeColor = 0x808080
oPivot.PivotColumns = "sum(5)[lower]/12"
oPivot.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatConditionalAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("lower","lower","")
   var_FormatConditionalAppearance.Expression = "len(value) != 0 ? (value < 100) :
0"
   var_FormatConditionalAppearance.ForeColor = 8421504
oPivot.PivotColumns = "sum(5)[lower]/12"
oPivot.EndUpdate()
```

Visual Objects

```
local var_FormatConditionalAppearance as IFormatConditionalAppearance

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:PivotRows := "0"
var_FormatConditionalAppearance :=
oDCOCX_Exontrol1:FormatConditionalAppearances:Add("lower","lower","")
   var_FormatConditionalAppearance:Expression := "len(value) != 0 ? (value < 100) :
0"
   var_FormatConditionalAppearance:ForeColor := RGB(128,128,128)
oDCOCX_Exontrol1:PivotColumns := "sum(5)[lower]/12"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oPivot,var_FormatConditionalAppearance

oPivot = ole_1.Object
oPivot.BeginUpdate()
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.PivotRows = "0"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Add("lower","lower","")
   var_FormatConditionalAppearance.Expression = "len(value) != 0 ? (value < 100) :
0"
   var_FormatConditionalAppearance.ForeColor = RGB(128,128,128)
oPivot.PivotColumns = "sum(5)[lower]/12"
oPivot.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
```

```
   Get ComImport "C:\Program Files\Exontrol\ExPivot\Sample\data.txt" Nothing to
Nothing
   Set ComPivotRows to "0"
   Variant voFormatConditionalAppearances
   Get ComFormatConditionalAppearances to voFormatConditionalAppearances
   Handle hoFormatConditionalAppearances
   Get Create (RefClass(cComFormatConditionalAppearances)) to
hoFormatConditionalAppearances
   Set pvComObject of hoFormatConditionalAppearances to
voFormatConditionalAppearances
      Variant voFormatConditionalAppearance
      Get ComAdd of hoFormatConditionalAppearances "lower" "lower" "" to
voFormatConditionalAppearance
      Handle hoFormatConditionalAppearance
      Get Create (RefClass(cComFormatConditionalAppearance)) to
hoFormatConditionalAppearance
      Set pvComObject of hoFormatConditionalAppearance to
voFormatConditionalAppearance
         Set ComExpression of hoFormatConditionalAppearance to "len(value) != 0 ?
(value < 100) : 0"
         Set ComForeColor of hoFormatConditionalAppearance to (RGB(128,128,128))
      Send Destroy to hoFormatConditionalAppearance
   Send Destroy to hoFormatConditionalAppearances
   Set ComPivotColumns to "sum(5)[lower]/12"
   Send ComEndUpdate
End_Procedure
```

XBase++

```
#include "AppEvent.ch"
#include "ActiveX.ch"

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oFormatConditionalAppearance
   LOCAL oPivot
```

```
   oForm := XbpDialog():new( AppDesktop() )
   oForm:drawingArea:clipChildren := .T.
   oForm:create( ,,{100,100}, {640,480},, .F. )
   oForm:close  := {|| PostAppEvent( xbeP_Quit )}

   oPivot := XbpActiveXControl():new( oForm:drawingArea )
   oPivot:CLSID  := "Exontrol.Pivot.1" /*{5C9DF3D3-81B1-42C4-BED6-
658F17748686}*/
   oPivot:create(,, {10,60},{610,370} )

      oPivot:BeginUpdate()
      oPivot:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
      oPivot:PivotRows := "0"
      oFormatConditionalAppearance :=
oPivot:FormatConditionalAppearances():Add("lower","lower","")
         oFormatConditionalAppearance:Expression := "len(value) != 0 ? (value < 100)
: 0"

oFormatConditionalAppearance:SetProperty("ForeColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 128,128,128 } )  , .F. ))
      oPivot:PivotColumns := "sum(5)[lower]/12"
      oPivot:EndUpdate()

   oForm:Show()
   DO WHILE nEvent != xbeP_Quit
      nEvent := AppEvent( @mp1, @mp2, @oXbp )
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO
RETURN
```

property FormatConditionalAppearance.Gradient as Color

Renders the text with a gradient color.

Type	Description
Color	A Color expression to define the second gradient color to end with.

By default, the Gradient property is 0, which indicates no effect. The GradientMode property specifies the way the gradient text is shown. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.GradientMode as Long

Indicates the gradient mode to be applied on text.

Type	Description
Long	A Long expression that indicates the way the gradient text is shown. The valid value is between 0 and 4.

By default, the GradientMode property is 1. The GradientMode property specifies the way the gradient text is shown. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.Italic as Boolean

Renders as italic text.

Type	Description
Boolean	A Boolean expression that specifies whether the text is rendered in italics.

By default, the Italic property is False. Use the Italic property on True, to show the objects in *italics*. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.Key as String

Indicates the key of the FormatConditionalAppearance object.

Type	Description
String	A String expression that specifies the unique key of the FormatConditionalAppearance object. The Key of the FormatConditionalAppearance should include only alpha-numeric characters, any other character will be ignored.

The Key property indicates the key of the FormatConditionalAppearance. *Use the Key of the FormatConditionalAppearance in [] to apply the specified object to any column or row, that matches the expression. For instance, the PivotRows property on "0[positive]" to group by the first column and shows the column's content in green for all positive values. The same rule is applied to PivotColumns or PivotTotals property.* The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

By default, the FormatConditionalAppearances collection contains the following keys:

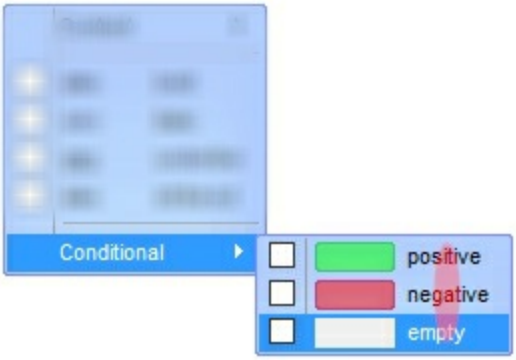
- **"positive",**
- **"negative"**

property FormatConditionalAppearance.Name as String

Specifies the name of the FormatConditionalAppearance object to be displayed on the context menu.

Type	Description
String	A String expression that defines the HTML caption to be displayed on the control's context menu.

The Name property indicates the name to be displayed on the control's context menu as shown bellow. The Key property specifies the key of the FormatConditionalAppearance object. The ToolTip property defines the FormatConditionalAppearance's tooltip. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.



The Name property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the

color in hexa values.

- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

  gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

  outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

  shadow

  or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

  outline anti-aliasing

property FormatConditionalAppearance.Outline as Color

Renders the text outlined with specified color.

Type	Description
Color	A Color expression that defines the color to show the outline text.

By default, the Outline property is 0, which indicates no effect. The OutlineSize property specifies the size of the outlined text/line. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.OutlineSize as Long

Indicates the size of the outline to be applied on text.

Type	Description
Long	A Long expression that defines the size of the outline text.

By default, the OutlineSize property is 1. If 0, the Outline has no effect. The OutlineSize property specifies the size of the outlined text/line. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.Shadow as Color

Renders the text with a shadow of specified color.

Type	Description
Color	A Color expression that defines the color of the shadow to be shown around the text

By default, the Shadow property is 0. If 0, this property has no effect. The ShadowOffset property defines the offset to show the text's shadow. The ShadowSize property define the size of the shadow to be shown. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.ShadowOffset as Long

Indicates the offset of the shadow to be applied on text.

Type	Description
Long	A Long expression that defines the offset where the text's shadow is shown.

By default, the ShadowOffset property is 2. If 0, this property shows the fine effect. The ShadowOffset property defines the offset to show the text's shadow. The ShadowColor property defines the color to show the text's shadow. The ShadowSize property define the size of the shadow to be shown. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.ShadowSize as Long

Indicates the size of the shadow to be applied on text.

Type	Description
Long	A Long expression that defines the size of the text's shadow.

By default, the ShadowSize property is 4. If 0, this property shows no shadow effect. The ShadowOffset property defines the offset to show the text's shadow. The ShadowColor property defines the color to show the text's shadow. The ShadowSize property define the size of the shadow to be shown. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.StrikeOut as Boolean

Specifies that the text should appear as strikeout.

Type	Description
Boolean	A Boolean expression that specifies whether the text is strikeout.

By default, the StrikeOut property is False. Use the StrikeOut property on True, to show the objects as ~~strikeout~~. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

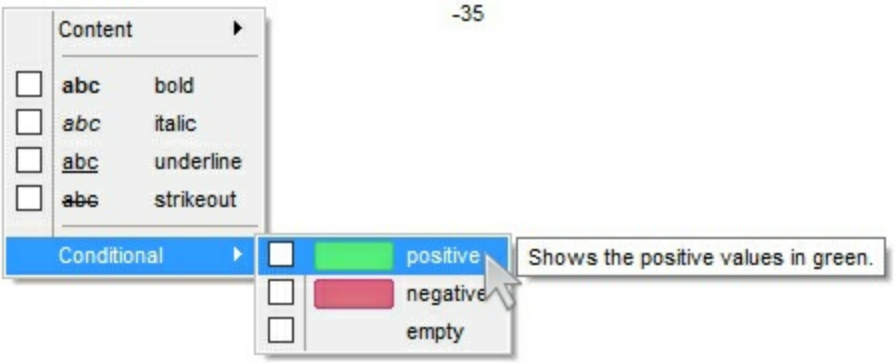
- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

property FormatConditionalAppearance.ToolTip as String

Specifies the tooltip of the FormatConditionalAppearance object to be displayed when the cursor hovers the object.

Type	Description
String	A String expression that defines the HTML caption to be shown when the cursor hovers the FormatConditionalAppearance object on the control's context menu.

The ToolTip property defines the FormatConditionalAppearance's tooltip. The Name property indicates the name to be displayed on the control's context menu. The Key property specifies the key of the FormatConditionalAppearance object. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.



The Name property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.

- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text

such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

property FormatConditionalAppearance.Underline as Boolean

Underlines the text.

Type	Description
Boolean	A Boolean expression that specifies whether the text is underlined.

By default, the Underline property is False. Use the Underline property on True, to show the objects underlined. The Expression property defines the conditional expression that determines whether the current format is applied on the pivot's cell based on the cell's value.

The following properties can be applied on objects:

- Bold, renders the text in bold (**bold**)
- Italic, renders the text in italic (*italic*)
- Underline, underlines the text (underline)
- StrikeOut, draws a line over the text. (~~strikeout~~)

Also, the decorative text is allowed as following:

- Shadow, shows the text with a shadow (shadow)
- Outline, shows the text as outlined (outlined)
- Gradient, shows the text in gradient (gradient)
- Fine, shows the text combined with a shadow around (fine)

FormatConditionalAppearances object

The FormatConditionalAppearances collection can be accessed through the control's FormatConditionalAppearances property. The FormatConditionalAppearance object changes the visual appearance of your data as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

By default, the control's context menu displays the following FormatConditionalAppearance objects:

By default, the FormatConditionalAppearances collection contains the following keys:

- **"negative",**
- **"positive"**

The FormatConditionalAppearances collection supports the following properties and methods:

Name	Description
Add	Adds a FormatConditionalAppearance object and returns a reference to the newly created object.
Clear	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
Item	Returns a specific FormatConditionalAppearance object giving its key.
Remove	Removes a specific member from the collection.

method FormatConditionalAppearances.Add (Key as String, [Name as Variant], [Expression as Variant])

Adds a FormatConditionalAppearance object and returns a reference to the newly created object.

Type	Description
Key as String	A String expression that specifies the unique key of the FormatConditionalAppearance object. The Key of the FormatConditionalAppearance should include only alpha-numeric characters, any other character will be ignored.
Name as Variant	A String expression that indicates the HTML caption to be displayed on the control's context menu.
Expression as Variant	A String expression that defines the conditional-expression to apply the current FormatConditionalAppearance. The Expression property defines the conditional-expression to apply the current format. For instance, "len(value) != 0" indicates any not-empty value.

Return	Description
FormatConditionalAppearance	A FormatConditionalAppearance object being created.

By default, the FormatConditionalAppearances collection contains the following keys: "positive", and "negative". The FormatConditionalAppearances helps you to provide conditional-format for your data, or in other words, ability to highlight values that matches a specified expression. Use the Clear method to remove all FormatConditionalAppearance objects from the FormatConditionalAppearances collection. Use the Remove method to remove a FormatConditionalAppearance object giving its key. Use the Item property to access a FormatConditionalAppearance object giving its key. The Key property indicates the key of the FormatConditionalAppearance object. The FormatAppearances collection provides formatting for entire column, no matter of what values it contains.

The following screen shows shows the negative/positive values with different colors/EBNs:

ShipCountry
ShipRegion

	Shippers_Co...		Shippers_Co...		
	Sum of Freight		Count of Freight		
					Total

Sum of Freight Count of Freight

ShipCountry	ShipRegion	United Package		Speedy Express		Federal Shipping	
Brazil	SP			108	1		
Canada	Québec					45	1
Finland		59	1				
France		-94	2			30	1
Germany		78	2	-45	1		
Ireland	Co. Cork					142	1
Mexico						-35	2
Poland		81	1				
Portugal		-13	1				
Switzerland		1	1				
USA	NM	-147	1				
	OR			-13	1		

By default, the FormatConditionalAppearances collection contains the following:

- "**positive**", shows positive values in green, while the expression is: "(dbl(value) != 0) ? (value > 0) : 0"
- "**negative**", shows negative values in red, while the expression is: "(dbl(value) != 0) ? (value < 0) : 0"

If the PivotBarVisible property includes the exPivotBarAllowFormatConditionalAppearance, the control's context menu include the "Conditional" item as shown bellow:

The FormatConditionalAppearance object changes the visual appearance of your values as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

method FormatConditionalAppearances.Clear ()

Removes all objects in a collection.

Type	Description

The Clear method removes all elements in the FormatConditionalAppearances collection. Excludes the exPivotBarAllowFormatConditionalAppearance flag from the PivotBarVisible property, and so no FormatConditionalAppearance objects are displayed on the control's context menu. The Remove method removes a FormatConditionalAppearance object giving its key. Use the Add method to add a new FormatConditionalAppearance object. You can use the **for each** statement to enumerate all objects in the FormatConditionalAppearances collection.

property FormatConditionalAppearances.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A Long expression that indicates the number of FormatConditionalAppearance objects in the FormatConditionalAppearances collection.

The Count property gets the number of FormatConditionalAppearance objects in the FormatConditionalAppearances collection. The Clear method removes all elements in the FormatConditionalAppearances collection. The Item property accesses a FormatConditionalAppearance object based on its key. Use the Add method to add a new FormatConditionalAppearance object. You can use the **for each** statement to enumerate all objects in the FormatConditionalAppearances collection.

The following samples show how to change the "positive" caption being displayed in the control's context menu.

VBA (MS Access, Excell...)

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatConditionalAppearances.Item("positive").Name = "Numere Positive"
End With
```

VB6

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatConditionalAppearances.Item("positive").Name = "Numere Positive"
End With
```

VB.NET

```
With Expivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatConditionalAppearances.Item("positive").Name = "Numere Positive"
End With
```

VB.NET for /COM

```
With AxPivot1
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .FormatConditionalAppearances.Item("positive").Name = "Numere Positive"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->GetFormatConditionalAppearances()->GetItem("positive")-
>PutName(L"Numere Positive");
```

C++ Builder

```
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->FormatConditionalAppearances->get_Item(TVariant("positive"))->Name =
L"Numere Positive";
```

C#

```
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.FormatConditionalAppearances["positive"].Name = "Numere Positive";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   Pivot1.FormatConditionalAppearances.Item("positive").Name = "Numere Positive";
</SCRIPT>
```

C# for /COM

```
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.FormatConditionalAppearances["positive"].Name = "Numere Positive";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatConditionalAppearance;
   anytype var_FormatConditionalAppearance;
   ;

   super();

   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   var_FormatConditionalAppearance =
COM::createFromObject(expivot1.FormatConditionalAppearances()).Item("positive");
com_FormatConditionalAppearance = var_FormatConditionalAppearance;
   com_FormatConditionalAppearance.Name("Numere Positive");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   FormatConditionalAppearances.Item['positive'].Name := 'Numere Positive';
end
```

Delphi (standard)

```
with Pivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
   FormatConditionalAppearances.Item['positive'].Name := 'Numere Positive';
end
```

VFP

```
with thisform.Pivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatConditionalAppearances.Item("positive").Name = "Numere Positive"
endwith
```

dBASE Plus

```
local oPivot,var_FormatConditionalAppearance

oPivot = form.Activex1.nativeObject
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
// oPivot.FormatConditionalAppearances.Item("positive").Name = "Numere
Positive"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Item("positive")
with (oPivot)
   TemplateDef = [Dim var_FormatConditionalAppearance]
   TemplateDef = var_FormatConditionalAppearance
   Template = [var_FormatConditionalAppearance.Name = "Numere Positive"]
endwith
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatConditionalAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
' oPivot.FormatConditionalAppearances.Item("positive").Name = "Numere
```

```
Positive"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Item("positive")
oPivot.TemplateDef = "Dim var_FormatConditionalAppearance"
oPivot.TemplateDef = var_FormatConditionalAppearance
oPivot.Template = "var_FormatConditionalAppearance.Name = \"Numere Positive\""
```

Visual Objects

```
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:FormatConditionalAppearances:[Item,"positive"]:Name :=
"Numere Positive"
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.FormatConditionalAppearances.Item("positive").Name = "Numere Positive"
```

property FormatConditionalAppearances.Item (Key as Variant) as FormatConditionalAppearance

Returns a specific FormatConditionalAppearance object giving its key.

Type	Description
Key as Variant	A String expression that specifies the key of the object to be retrieved.
FormatConditionalAppearance	A FormatConditionalAppearance object being requested.

The Item property accesses a FormatConditionalAppearance object based on its key. Use the Add method to add a new FormatConditionalAppearance object. The Count property gets the number of FormatConditionalAppearance objects in the FormatConditionalAppearances collection. You can use the **for each** statement to enumerate all objects in the FormatConditionalAppearances collection.

The following samples show how to change the "negative" caption being displayed in the control's context menu.

VBA (MS Access, Excell...)

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatConditionalAppearances.Item("negative").Name = "Numere Negative"
End With
```

VB6

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatConditionalAppearances.Item("negative").Name = "Numere Negative"
End With
```

VB.NET

```
With Expivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatConditionalAppearances.Item("negative").Name = "Numere Negative"
End With
```

VB.NET for /COM

```
With AxPivot1
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .FormatConditionalAppearances.Item("negative").Name = "Numere Negative"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->GetFormatConditionalAppearances()->GetItem("negative")-
>PutName(L"Numere Negative");
```

C++ Builder

```
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->FormatConditionalAppearances->get_Item(TVariant("negative"))->Name =
L"Numere Negative";
```

C#

```
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.FormatConditionalAppearances["negative"].Name = "Numere Negative";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   Pivot1.FormatConditionalAppearances.Item("negative").Name = "Numere
Negative";
</SCRIPT>
```

C# for /COM

```
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.FormatConditionalAppearances["negative"].Name = "Numere Negative";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_FormatConditionalAppearance;
   anytype var_FormatConditionalAppearance;
   ;

   super();

   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   var_FormatConditionalAppearance =
COM::createFromObject(expivot1.FormatConditionalAppearances()).Item("negative");
com_FormatConditionalAppearance = var_FormatConditionalAppearance;
   com_FormatConditionalAppearance.Name("Numere Negative");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   FormatConditionalAppearances.Item['negative'].Name := 'Numere Negative';
end
```

Delphi (standard)

```
with Pivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
   FormatConditionalAppearances.Item['negative'].Name := 'Numere Negative';
end
```

VFP

```
with thisform.Pivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatConditionalAppearances.Item("negative").Name = "Numere Negative"
endwith
```

dBASE Plus

```
local oPivot,var_FormatConditionalAppearance

oPivot = form.Activex1.nativeObject
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
// oPivot.FormatConditionalAppearances.Item("negative").Name = "Numere
Negative"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Item("negative")
with (oPivot)
   TemplateDef = [Dim var_FormatConditionalAppearance]
   TemplateDef = var_FormatConditionalAppearance
   Template = [var_FormatConditionalAppearance.Name = "Numere Negative"]
endwith
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatConditionalAppearance as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
' oPivot.FormatConditionalAppearances.Item("negative").Name = "Numere
```

```
Negative"
var_FormatConditionalAppearance =
oPivot.FormatConditionalAppearances.Item("negative")
oPivot.TemplateDef = "Dim var_FormatConditionalAppearance"
oPivot.TemplateDef = var_FormatConditionalAppearance
oPivot.Template = "var_FormatConditionalAppearance.Name = \"Numere
Negative\""
```

Visual Objects

```
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:FormatConditionalAppearances:[Item,"negative"]:Name :=
"Numere Negative"
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.FormatConditionalAppearances.Item("negative").Name = "Numere Negative"
```

method FormatConditionalAppearances.Remove (Key as Variant)

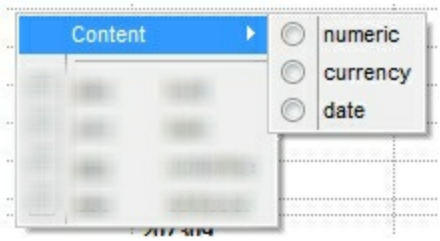
Removes a specific member from the collection.

Type	Description
Key as Variant	A String expression that indicates the Key of the FormatConditionalAppearance object to be removed.

The Remove method removes a FormatConditionalAppearance object giving its key. The Clear method removes all elements in the FormatConditionalAppearances collection. Excludes the exPivotBarAllowFormatConditionalAppearance flag from the PivotBarVisible property, and so no FormatConditionalAppearance objects are displayed on the control's context menu. Use the Add method to add a new FormatConditionalAppearance object. You can use the **for each** statement to enumerate all objects in the FormatConditionalAppearances collection.

FormatContent object

The FormatContent object holds information about how a column or row can be displayed, formatted or converted. For instance, you need to display a column in numeric format with grouping by digits, or as date in long format, and so on. Each FormatContent object is shown in the control's context menu under the Content submenu as radio buttons as shown bellow:



The FormatContent object supports the following properties and methods.

Name	Description
Expression	Specifies the expression format the content of the column or row.
Key	Indicates the key of the FormatContent object.
Name	Specifies the name of the FormatContent object to be displayed on the context menu.
ToolTip	Specifies the tooltip of the FormatContent object to be displayed when the cursor hovers the object.

property FormatContent.Expression as String

Specifies the expression format the content of the column or row.

Type	Description
String	A String expression that defines the format to be shown

The Expression property indicates the format to be displayed instead the value itself. If the Expression is not valid, the FormatContent object shows as disabled in the control's context menu. The Name property specifies the HTML caption to be displayed on the control's context menu. The ToolTip property specifies the HTML tooltip to be displayed on the control's context menu, when the cursor hovers the object.

For instance:

- *"len(value) ? (value format ") : ""*, displays the value in numeric format, if it is not empty.
- *"len(value) ? currency(value) :""*, displays the value in currency format, if it is not empty.
- *"upper(value)"*, displays the value in upper-case.

The **value** keyword in the Expression property indicates the value/result to be converted.

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- ***/*** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

The supported binary operators, all these with the same priority 0, are :

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore

any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for *?* operator is

  expression ? true_part : false_part

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

  expression array (c1,c2,c3,...cn)

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')* is equivalent with *month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'O';10:'N';11:'D')*.

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

expression in (c1,c2,c3,...cn)

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *value in (11,22,33,44,13)* is equivalent with *(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

expression switch (default,c1,c2,c3,...,cn)

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the *%0 switch ('not found',1,4,7,9,11)* gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates

or strings. For instance, the *date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)* indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: *date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)* statement indicates the working hours for dates as follows:

- #4/1/2009#, from hours 06:00 AM to 12:00 PM
- #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
- #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions. Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance *type(%1) = 8* specifies the cells (on the column 1) that contains string values.

  Here's few predefined types:

  - 0 - empty (not initialized)
  - 1 - null
  - 2 - short
  - 3 - long
  - 4 - float
  - 5 - double
  - 6 - currency
  - 7 - date
  - 8 - string
  - 9 - object
  - 10 - error
  - 11 - boolean
  - 12 - variant
  - 13 - any
  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long

  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the *1000 format ''* displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no

formatting.

The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as

appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"

- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- a **split** b, splits the a using the separator b, and returns an array. For instance, the *weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' split ' '* gets the weekday as string. This operator can be used with the array.

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"

- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

property FormatContent.Key as String

Indicates the key of the FormatContent object.

Type	Description
String	A String expression that defines the key of the FormatContent object. The Key must includes alpha-numeric characters, any other character is ignored.

The Key property specifies the key of the FormatContent object. The Expression property defines the format to be applied. The Name property specifies the HTML caption to be displayed on the control's context menu. The ToolTip property specifies the HTML tooltip to be displayed on the control's context menu, when the cursor hovers the object.

By default, the FormatContents collection contains the following keys:

- **"numeric"**, with the Expression as "len(value) ? (value format '') : ""
- **"currency"**, with the Expression as "len(value) ? currency(value) :""
- **"date"**, with the Expression as "date(dbl(value))"

property FormatContent.Name as String

Specifies the name of the FormatContent object to be displayed on the context menu.

Type	Description
String	A String expression that defines the HTML caption to be displayed on the control's context menu.

The Name property defines the HTML caption to be displayed on the control's context menu. You can use the <img> HTML tag to display icons or pictures. Use the Images or HTMLPicture method to add icons or pictures to control. The ToolTip property specifies the HTML tooltip to be displayed on the control's context menu, when the cursor hovers the object.

The following screen shot shows the Content sub menu:

The Name property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text

with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the

red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

property FormatContent.ToolTip as String

Specifies the tooltip of the FormatContent object to be displayed when the cursor hovers the object.

Type	Description
String	A String expression that defines the HTML tooltip to be shown when the cursor hovers the object in the control's context menu.

The ToolTip property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires

<solidline> or <dotline>).

- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines

the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

FormatContents object

The FormatContents object holds a collection of FormatContent objects. The FormatContent object holds information about how a column or row can be displayed, formatted or converted. The FormatContents property accesses the FormatContents collection. The following sample shows the control's context menu with a different appearance and more converting functions:

If the PivotBarVisible property includes the exPivotBarAllowFormatContent flag, the control's context menu includes the FormatContent objects. If the exPivotBarAllowFormatContent flag is missing from the PivotBarVisible property, the control's context menu displays no FormatContent objects.

By default, the FormatContents collection contains the following keys:

- **"numeric"**, with the Expression as "len(value) ? (value format '') : ""
- **"currency"**, with the Expression as "len(value) ? currency(value) :""
- **"date"**, with the Expression as "date(dbl(value))"

The FormatContents collection supports the following properties and methods:

Name	Description
Add	Adds a FormatContent object and returns a reference to the newly created object.
Clear	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
Item	Returns a specific FormatContent object giving its key.
Remove	Removes a specific member from the collection.

method FormatContents.Add (Key as String, Expression as String, [Name as Variant])

Adds a FormatContent object and returns a reference to the newly created object.

Type	Description
Key as String	A String expression that defines the unique key to identify the FormatContent object. The Key must includes alpha-numeric characters, any other character will be ignored.
Expression as String	The Expression parameter defines the format to be applied on the column or row. If the Expression is not valid, the FormatContent objects shows as disabled on the control's context menu. The **value** keyword defines the value to be converted, and the Expression supports a lot of predefined functions as shown here.
Name as Variant	A String expression that defines the HTML Caption to be shown on the control's context menu.

Return	Description
FormatContent	A FormatContent object being created.

The Add method adds a new FormatContent object to the collection. The Expression property defines the format to be applied on the column or row. If the Expression is not valid, the FormatContent objects shows as disabled on the control's context menu. The ToolTip property defines the FormatContent's tooltip, which is shown when the cursor hovers the object in the control's context menu.

Here's a few samples on how to use the FormatContent objects:

- *FormatContents.Add("upper","upper(value)")*, displays the column/row in upper-case, such as 'ROMANIA' instead 'Romania'
- *FormatContents.Add("longdate","longdate(date(value))")*, displays the object's content as date in long format, such as 'Monday, December 31, 2012'
- *FormatContents.Add("letter","'<fgcolor=808080>' + upper(value left 1) + '</fgcolor> ' + value")*, shows the first letter twice in bold and gray, such as 'R

Romania' instead 'romania'

- *FormatContents.Add("proper","'' + ((0:=proper(value)) left 1) + '' + (=:0 mid 2)")*, displays the first letter in bold and upper-case, and let the rest unchanged, such as '**M**ihai Filimon' instead 'mihai filimon'.

Use the Key of the FormatContent in [content=key] to apply the specified object to any column or row. For instance, the PivotRows property on "0[content=numeric]" to group by the first column and shows the column's content as numeric. The same rule is applied to PivotColumns or PivotTotals property.

The following samples show how you can display the column in upper-case.

VBA (MS Access, Excell...)

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatContents.Add "upper","upper(value)"
   .PivotRows = "0[content=upper]"
End With
```

VB6

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .FormatContents.Add "upper","upper(value)"
   .PivotRows = "0[content=upper]"
End With
```

VB.NET

```
With Expivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatContents.Add("upper","upper(value)")
   .PivotRows = "0[content=upper]"
End With
```

VB.NET for /COM

```
With AxPivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
```

```
    .FormatContents.Add("upper","upper(value)")
    .PivotRows = "0[content=upper]"
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->GetFormatContents()->Add(L"upper",L"upper(value)",vtMissing);
spPivot1->PutPivotRows(L"0[content=upper]");
```

C++ Builder

```
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->FormatContents->Add(L"upper",L"upper(value)",TNoParam());
Pivot1->PivotRows = L"0[content=upper]";
```

C#

```
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.FormatContents.Add("upper","upper(value)",null);
expivot1.PivotRows = "0[content=upper]";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   Pivot1.FormatContents.Add("upper","upper(value)",null);
   Pivot1.PivotRows = "0[content=upper]";
</SCRIPT>
```

C# for /COM

```
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.FormatContents.Add("upper","upper(value)",null);
axPivot1.PivotRows = "0[content=upper]";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   ;

   super();

   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   expivot1.FormatContents().Add("upper","upper(value)");
   expivot1.PivotRows("0[content=upper]");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
   FormatContents.Add('upper','upper(value)',Nil);
   PivotRows := '0[content=upper]';
end
```

Delphi (standard)

```
with Pivot1 do
```

```
begin
   Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
   FormatContents.Add('upper','upper(value)',Null);
   PivotRows := '0[content=upper]';
end
```

VFP

```
with thisform.Pivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   .FormatContents.Add("upper","upper(value)")
   .PivotRows = "0[content=upper]"
endwith
```

dBASE Plus

```
local oPivot

oPivot = form.Activex1.nativeObject
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.FormatContents.Add("upper","upper(value)")
oPivot.PivotRows = "0[content=upper]"
```

XBasic (Alpha Five)

```
Dim oPivot as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.FormatContents.Add("upper","upper(value)")
oPivot.PivotRows = "0[content=upper]"
```

Visual Objects

```
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:FormatContents:Add("upper","upper(value)",nil)
```

```
oDCOCX_Exontrol1:PivotRows := "0[content=upper]"
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.FormatContents.Add("upper","upper(value)")
oPivot.PivotRows = "0[content=upper]"
```

method FormatContents.Clear ()

Removes all objects in a collection.

Type	Description

The Clear method clears all FormatContent objects from the collection. The Remove method removes a FormatContent object based on its key. If the PivotBarVisible property includes the exPivotBarAllowFormatContent flag, the control's context menu includes the FormatContent objects. If the exPivotBarAllowFormatContent flag is missing from the PivotBarVisible property, the control's context menu displays no FormatContent objects. The Count property specifies the number of FormatContent objects in the collection. The Item property accesses a FormatContent object based on its key.

property FormatContents.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A Long expression that specifies the number of FormatContent objects.

The Count property specifies the number of FormatContent objects in the collection. The Item property accesses a FormatContent object based on its key. You can use the **for each** statement to enumerate the FormatContent objects in the FormatContents collection.

property FormatContents.Item (Key as Variant) as FormatContent

Returns a specific FormatContent object giving its key.

Type	Description
Key as Variant	A String expression that indicates the Key of the FormatContent to be accessed.
FormatContent	A FormatContent object being requested.

The Item property accesses a FormatContent object based on its key. The Count property specifies the number of FormatContent objects in the collection. You can use the **for each** statement to enumerate the FormatContent objects in the FormatContents collection. You can use the <img> HTML tag to display icons or pictures. Use the Images or HTMLPicture method to add icons or pictures to control.

The following samples show how to assign an icon/image/picture to a FormatContent object being shown in the control's context menu:

VBA (MS Access, Excell...)

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vr
 & _
"/oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
   .PivotRows = "5[content=numeric]"
End With
```

VB6

```
With Pivot1
   .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
```

```
    .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vr
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
    .FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
    .PivotRows = "5[content=numeric]"
End With
```

VB.NET

```
With Expivot1
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")

.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0o
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=")
    .FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
    .PivotRows = "5[content=numeric]"
End With
```

VB.NET for /COM

```
With AxPivot1
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")

.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0o
```

```
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=")
   .FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
   .PivotRows = "5[content=numeric]"
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

   #import <ExPivot.dll>
   using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
spPivot1->GetFormatContents()->GetItem("numeric")->PutName(L"
<img>1</img> Numeric");
spPivot1->PutPivotRows(L"5[content=numeric]");
```

C++ Builder

```
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1-
>Images(TVariant(String("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaI
 +
"/oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="));
Pivot1->FormatContents->get_Item(TVariant("numeric"))->Name = L"
<img>1</img> Numeric";
Pivot1->PivotRows = L"5[content=numeric]";
```

C#

```
expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAI
 +
"/oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
expivot1.FormatContents["numeric"].Name = "<img>1</img> Numeric";
expivot1.PivotRows = "5[content=numeric]";
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);

Pivot1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAk
 +

"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 +

"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 +

"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 +
   "NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
   Pivot1.FormatContents.Item("numeric").Name = "<img>1</img> Numeric";
   Pivot1.PivotRows = "5[content=numeric]";
</SCRIPT>
```

C# for /COM

```
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAI
 +
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
axPivot1.FormatContents["numeric"].Name = "<img>1</img> Numeric";
axPivot1.PivotRows = "5[content=numeric]";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_FormatContent;
    anytype var_FormatContent;
    str var_s;
    ;

    super();

    expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
    var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vr

    var_s = var_s +
"oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxu\

    var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0

    var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB0

    var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=";
    expivot1.Images(COMVariant::createFromStr(var_s));
    var_FormatContent =
COM::createFromObject(expivot1.FormatContents()).Item("numeric");
com_FormatContent = var_FormatContent;
    com_FormatContent.Name("<img>1</img> Numeric");
    expivot1.PivotRows("5[content=numeric]");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
```

```
  Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);

Images('gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olU
 +

'oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu\
 +

'wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0
 +

'3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB0
 +
  'AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=');
  FormatContents.Item['numeric'].Name := '<img>1</img> Numeric';
  PivotRows := '5[content=numeric]';
end
```

Delphi (standard)

```
with Pivot1 do
begin
  Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);

Images('gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olU
 +

'oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxu\
 +

'wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0
 +

'3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB0
 +
  'AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=');
  FormatContents.Item['numeric'].Name := '<img>1</img> Numeric';
```

```
   PivotRows := '5[content=numeric]';
end
```

VFP

```
with thisform.Pivot1
   .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vr

   var_s = var_s +
"oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxu

   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vI8j4f/qfEZeB0

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   .FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
   .PivotRows = "5[content=numeric]"
endwith
```

dBASE Plus

```
local oPivot,var_FormatContent

oPivot = form.Activex1.nativeObject
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAk

// oPivot.FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
var_FormatContent = oPivot.FormatContents.Item("numeric")
with (oPivot)
   TemplateDef = [Dim var_FormatContent]
```

```
    TemplateDef = var_FormatContent
    Template = [var_FormatContent.Name = "<img>1</img> Numeric"]
endwith
oPivot.PivotRows = "5[content=numeric]"
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim var_FormatContent as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAk

' oPivot.FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
var_FormatContent = oPivot.FormatContents.Item("numeric")
oPivot.TemplateDef = "Dim var_FormatContent"
oPivot.TemplateDef = var_FormatContent
oPivot.Template = "var_FormatContent.Name = \"<img>1</img> Numeric\""

oPivot.PivotRows = "5[content=numeric]"
```

Visual Objects

```
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZF

oDCOCX_Exontrol1:FormatContents:[Item,"numeric"]:Name := "<img>1</img>
Numeric"
oDCOCX_Exontrol1:PivotRows := "5[content=numeric]"
```

PowerBuilder

```
OleObject oPivot
```

```
oPivot = ole_1.Object
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAk

oPivot.FormatContents.Item("numeric").Name = "<img>1</img> Numeric"
oPivot.PivotRows = "5[content=numeric]"
```

method FormatContents.Remove (Key as Variant)

Removes a specific member from the collection.

Type	Description
Key as Variant	A String expression that defines the key of the FormatContent object to be removed.

The Remove method removes a FormatContent object based on its key. The Clear method clears all FormatContent objects from the collection. If the PivotBarVisible property includes the exPivotBarAllowFormatContent flag, the control's context menu includes the FormatContent objects. If the exPivotBarAllowFormatContent flag is missing from the PivotBarVisible property, the control's context menu displays no FormatContent objects. The Count property specifies the number of FormatContent objects in the collection. The Item property accesses a FormatContent object based on its key.

Pivot object

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {5C9DF3D3-81B1-42C4-BED6-658F17748686}. The object's program identifier is: "Exontrol.Pivot". The /COM object module is: "ExPivot.dll"

The Exontrol's eXPivot tool is our approach to provide data summarization, as a pivot table. A pivot-table can automatically sort, count, total or give the average of the data stored in one table or spreadsheet. The user sets up and changes the summary's structure by dragging and dropping fields graphically.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method

The AppendData method appends data to the control (prevents clearing data already loaded).

The following properties may be used to group and summarize the data, once it is loaded:

- PivotRows property specifies the list of DATA columns that determines the first column in the control's list. In other words, the Group-By columns
- PivotColumns property specifies the list of DATA columns that determines the rest of the columns to be displayed on the control's list
- PivotTotals property specifies the list of total/sub-total functions to be displayed on the control's list.

The Pivot object supports the following properties and methods:

Name	Description
Aggregates	Retrieves the Aggregates collection of the pivot control.
AllowDrop	Gets or sets a value indicating whether the control can be used as the target of a drag-and-drop operation.
AllowSelectNothing	Specifies whether the current selection is erased, once the user clicks outside of the items section.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
AppendData	Appends data to the control (prevents clearing data already loaded).
	Attaches a script to the current object, including the

AttachTemplate	events, from a string, file, a safe array of bytes.
AutoDrag	Gets or sets a value that indicates the way the component supports the AutoDrag feature.
BackColor	Specifies the control's background color.
BackColorAlternate	Specifies the background color used to display alternate items in the control.
BackColorHeader	Specifies the header's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BorderHeight	Sets or retrieves a value that indicates the border height of the control.
BorderWidth	Sets or retrieves a value that indicates the border width of the control.
CheckImage	Retrieves or sets a value that indicates the image used by cells of checkbox type.
ClearData	Removes the control's data.
ClearFilter	Clears the filter.
CollapseAll	Collapses all rows.
ColumnAutoResize	Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.
ColumnFromPoint	Retrieves the column from the point.
Copy	Copies the control's content to the clipboard, in the EMF format.
CopyTo	Exports the control's view to an EMF file.
DataColumnFromPoint	Retrieves the index of the data column from the point.
DataColumns	Retrieves the Data Columns collection of the pivot control.
DataSource	Retrieves or sets a value that indicates the data source for object.
DefaultColumnWidth	Retrieves or sets a value that indicates the default column width.

DefaultItemHeight	Retrieves or sets a value that indicates the default item height.
Description	Changes descriptions for control objects.
DisplayFilterList	Specifies what the column's filter displays.
DisplayPivotData	Retrieves or sets the maximum number of rows to be displayed on the control's list.
DisplayPivotFields	Retrieves or sets the maximum number of columns to be displayed on the control's list.
DisplayPivotRows	Retrieves or sets the maximum number of rows to be generated on the control's list.
DrawGridLines	Retrieves or sets a value that indicates whether the grid lines are visible or hidden.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
ExpandAll	Expands all rows.
ExpandOnDblClick	Specifies whether the item is expanded or collapsed if the user dbl clicks the item.
Export	Exports the control's data to a CSV format.
FilterBarBackColor	Specifies the background color of the control's filter bar.
FilterBarCaption	Specifies the filter bar's caption.
FilterBarFont	Retrieves or sets the font for control's filter bar.
FilterBarForeColor	Specifies the foreground color of the control's filter bar.
FilterBarHeight	Specifies the height of the control's filter bar. If the value is less than 0, the filterbar is automatically resized to fit its description.
FilterBarPrompt	Specifies the caption to be displayed when the filter pattern is missing.
FilterBarPromptColumns	Specifies the list of columns to be used when filtering using the prompt.
FilterBarPromptPattern	Specifies the pattern for the filter prompt.
FilterBarPromptType	Specifies the type of the filter prompt.

FilterBarPromptVisible	Shows or hides the filter prompt.
FilterCriteria	Retrieves or sets the filter criteria.
FilterInclude	Specifies the items being included after the user applies the filter.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
ForeColorHeader	Specifies the header's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
FormatAppearances	Retrieves the FormatAppearances collection of the pivot control.
FormatConditionalAppearances	Retrieves the FormatConditionalAppearances collection of the pivot control.
FormatContents	Retrieves the FormatContents collection of the pivot control.
FormatPivotAggregate	Specifies the format to display an aggregate function.
FormatPivotHeader	Specifies the format to display the columns in the pivot bar.
FormatPivotTotal	Specifies the format to display an aggregate/total functions.
GetHeaders	Gets a safe array of all generated columns/headers.
GetItems	Gets a safe array of all generated items/values.
GridLineColor	Specifies the grid line color.
GridLineStyle	Specifies the style for gridlines in the list part of the control.
HasLines	Enhances the graphic representation of a grid control's hierarchy by drawing lines that link child items to their corresponding parent item.
HeaderAppearance	Retrieves or sets a value that indicates the header's appearance.
HeaderHeight	Retrieves or sets a value indicating the control's header height.
HeaderVisible	Retrieves or sets a value that indicates whether the the

	grid's header is visible or hidden.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Images List Control.
ImageSize	Retrieves or sets the size of icons the control displays.
Import	Imports the control's data from a CSV format.
IncludeExpandColumn	Specifies whether the column itself is displayed in the list (header/chart), while it expanded (the column contains child columns).
Indent	Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
Layout	Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.
LinesAtRoot	Link items at the root of the hierarchy.
LoadHeadersOnly	Loads the headers only, so no data is loaded.
LoadXML	Loads an XML document from the specified location, using MSXML parser.
LockRowsColumn	Retrieves or sets a value that indicates whether the rows column in the list is locked or scrollable.
LockTotalRows	Retrieves or sets a value that indicates whether the total rows in the list are locked or scrollable.
OnFilterChange	Specifies the action that the control performs once the user changes the filter at runtime.
PaneHeight	Specifies the height for the top or bottom panel.
PaneMinHeight	Specifies the minimum height for the top or bottom panel.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
PivotBarVisible	Specifies how the control displays its pivot bar.
PivotColumns	Specifies the list of columns to be displayed in the list.
PivotColumnsFloatBarVisible	Retrieves or sets a value that indicates whether the pivot columns float bar is visible or hidden.
PivotColumnsSortOrder	Specifies the sorting order for the columns being shown in

	the control's columns floating panel.
PivotRows	Specifies the list of group-by columns that determines the rows in the list.
PivotTotalDefaultFormatAppearances	Specifies the list of format-appearances (key of FormatAppearance object), separated by comma, to be applied on the Total field when it is displayed in the pivot-table.
PivotTotalDefaultFormatContent	Specifies the default format (key of FormatContent object) to be applied on the Total field when it is displayed in the pivot-table.
PivotTotals	Indicates the list of totals/subtotals to be shown in the list.
RadioImage	Retrieves or sets a value that indicates the image used by cells of radio type.
Refresh	Refreses the control.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
Reset	Resets the control's layouts, so no filtering, sorting, ... is applied to the view.
SaveXML	Saves the control's content as XML document to the specified location, using the MSXML parser.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
SelBackMode	Retrieves or sets a value that indicates whether the selection is transparent or opaque.
SelectableAggregateRows	Specifies whether the aggregate rows are selectable or un-selectable.
SelectAll	Selects all rows.
SelectOnRelease	Indicates whether the selection occurs when the user releases the mouse button.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowBranchRows	Indicates how the branch rows displays the information (divider items).
ShowDataOnDblClick	Specifies whether the user shows the original data that generated the result when user double clicks a cell.
ShowImageList	Specifies whether the control's image list window is visible

	or hidden.
ShowToolTip	Shows the specified tooltip at given position.
ShowViewCompact	Indicates whether the view compacts the data being displayed.
SingleSel	Retrieves or sets a value that indicates whether the control supports single or multiple selection.
Statistics	Gives statistics data of objects being hold by the control.
Template	Specifies the control's template.
TemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.
ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipFont	Retrieves or sets the tooltip's font.
ToolTipPopDelay	Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
UnselectAll	Unselects all rows.
UseVisualTheme	Specifies whether the control uses the current visual theme to display certain UI parts.
ValueFromPoint	Retrieves the value from the point.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.
VisualDesign	Invokes the control's VisualAppearance designer.

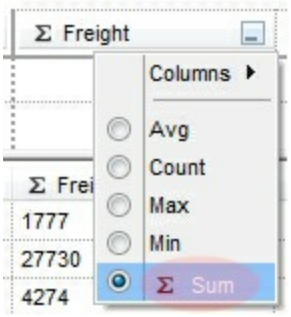
property Pivot.Aggregates as Aggregates

Retrieves the Aggregates collection of the pivot control.

Type	Description
Aggregates	The Aggregates collection that holds the Aggregate objects.

The Aggregates property gives access to the Aggregates collection. The Aggregates collection holds a collection of Aggregate objects. An aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning or measurement such as a set, a bag or a list. The Aggregate object is identified by an unique key. By default, the Aggregates collection contains the "sum", "min", "max", "count" or "avg" Aggregate objects. Use the Add method to add a new Aggregate function/object to the control. Use the FormatValue/FormatResult property to convert/format the value/result of the Aggregate function. The PivotColumns or PivotTotals property to display aggregate functions on the columns. Use the FormatPivotAggregate / FormatPivotTotal property to display aggregate functions in a different format.

The following screen shot shows the Aggregate objects in the control's context menu:



By default, the Aggregates collection contains the following elements:

- **"sum"**, summation is the operation of adding a sequence of numbers; the result is their sum or total
- **"min"**, minimum is the smallest value
- **"max"**, maximum is the largest value
- **"count"**, counts the number of objects in the set
- **"avg"**, average is the arithmetic mean, which means sum of all numbers divided by the count.

For instance:

- you want to get the total for negative values or to count the positive value only. In this case, you can add a new Aggregate object such as Aggregates.Add("negative",

"sum").FormatValue = "value < 0 ? value : 0", and so the negative Aggregate function gets the total of negative values only.

- The Aggregates.Add("positive", "sum").FormatValue = "value < 0 ? 0 : 1", counts the number of positive values.

property Pivot.AllowDrop as Boolean

Gets or sets a value indicating whether the control can be used as the target of a drag-and-drop operation.

Type	Description
Boolean	A Boolean expression that indicates whether the control allows OLE Drag and Drop data.

By default, the AllowDrop property is True. Use the AllowDrop property on False, to prevent loading the data-files (TXT, XML files), by drag and drop, into the control. Use the AllowGroupBy property to specify whether the user can group by specified column. The Import/DataSource/LoadXML method loads programmatically data to the control.

property Pivot.AllowSelectNothing as Boolean

Specifies whether the current selection is erased, once the user clicks outside of the items section.

Type	Description
Boolean	A Boolean expression that specifies whether the current selection is erased, once the user clicks outside of the items section.

By default, the AllowSelectNothing property is False. The AllowSelectNothing property specifies whether the current selection is erased, once the user clicks outside of the items section. For instance, if the control's SingleSel property is True, and AllowSelectNothing property is True, you can un-select the single-selected item if pressing the CTRL + Space, or by CTRL + click.

property Pivot.AnchorFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as String

Retrieves the identifier of the anchor from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
String	A String expression that specifies the identifier (id) of the anchor element from the point, or empty string if there is no anchor element at the cursor.

Use the AnchorFromPoint property to determine the identifier of the anchor from the point. Use the <a id;options> anchor elements to add hyperlinks to cell's caption. The control fires the AnchorClick event when the user clicks an anchor element. Use the ShowToolTip method to show the specified tooltip at given or cursor coordinates. The MouseMove event is generated continually as the mouse pointer moves across the control.

The following VB sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub Pivot1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   With Pivot1
      .ShowToolTip .AnchorFromPoint(-1, -1)
   End With
End Sub
```

The following VB.NET sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub AxPivot1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_MouseMoveEvent) Handles AxPivot1.MouseMoveEvent
   With AxPivot1
      .ShowToolTip(.get_AnchorFromPoint(-1, -1))
   End With
End Sub
```

The following C# sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
private void axPivot1_MouseMoveEvent(object sender,
AxEXPIVOTLib._IPivotEvents_MouseMoveEvent e)
{
   axPivot1.ShowToolTip(axPivot1.get_AnchorFromPoint(-1, -1));
}
```

The following C++ sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
void OnMouseMovePivot1(short Button, short Shift, long X, long Y)
{
   COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
   m_pivot.ShowToolTip( m_pivot.GetAnchorFromPoint( -1, -1 ), vtEmpty, vtEmpty,
vtEmpty );
}
```

The following VFP sample displays (as tooltip) the identifier of the anchor element from the cursor:

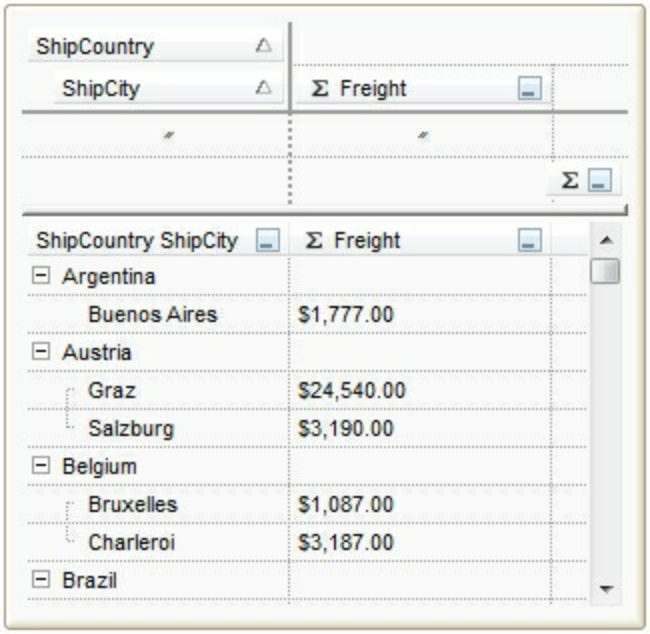
```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform
   With .Pivot1
      .ShowToolTip(.AnchorFromPoint(-1, -1))
   EndWith
endwith
```

property Pivot.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the control's appearance, or a color expression whose last 7 bits in the high significant byte of the value indicates the index of the skin in the Appearance collection, being displayed as control's borders. For instance, if the Appearance = 0x1000000, indicates that the first skin object in the Appearance collection defines the control's border. ***The Client object in the skin, defines the client area of the control. The list/hierarchy/chart, scrollbars are always shown in the control's client area. The skin may contain transparent objects, and so you can define round corners. The normal.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file***

Use the Appearance property to specify the control's border. Use the Add method to add new skins to the control. Use the BackColor property to specify the control's background color. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips.



The following VB sample changes the visual aspect of the borders of the control (please check the above picture for round corners):

```
With Pivot1
```

```
    .BeginUpdate
        .VisualAppearance.Add &H16, "c:\temp\normal.ebn"
        .Appearance = &H16000000
        .BackColor = RGB(250, 250, 250)
    .EndUpdate
End With
```

The following VB.NET sample changes the visual aspect of the borders of the control:

```
With AxPivot1
    .BeginUpdate()
    .VisualAppearance.Add(&H16, "c:\temp\normal.ebn")
    .Appearance = &H16000000
    .BackColor = Color.FromArgb(250, 250, 250)
    .EndUpdate()
End With
```

The following C# sample changes the visual aspect of the borders of the control:

```
axPivot1.BeginUpdate();
axPivot1.VisualAppearance.Add(0x16, "c:\\temp\\normal.ebn");
axPivot1.Appearance = (EXPIVOTLib.AppearanceEnum)0x16000000;
axPivot1.BackColor = Color.FromArgb(250, 250, 250);
axPivot1.EndUpdate();
```

The following C++ sample changes the visual aspect of the borders of the control:

```
m_pivot.BeginUpdate();
m_pivot.GetVisualAppearance().Add( 0x16, COleVariant( "c:\\temp\\normal.ebn" ) );
m_pivot.SetAppearance( 0x16000000 );
m_pivot.SetBackColor( RGB(250,250,250) );
m_pivot.EndUpdate();
```

The following VFP sample changes the visual aspect of the borders of the control:

```
with thisform.Pivot1
    .BeginUpdate
        .VisualAppearance.Add(0x16, "c:\temp\normal.ebn")
        .Appearance = 0x16000000
```

```
        .BackColor = RGB(250, 250, 250)
    .EndUpdate
endwith
```

method Pivot.AppendData ([Source as Variant], [Options as Variant])

Appends data to the control (prevents clearing data already loaded).

Type	Description
Source as Variant	Indicates the data to append as one of the following: • ADO.Recordset, ADODB.Recordset or DAO recordset (similar with DataSource, Options parameter has no effect) • An indicator of the object that specifies the source for the XML document. The object can represent a file name, a URL, an IStream, a SAFEARRAY, or an IXMLDOMDocument (similar with LoadXML, Options parameter has no effect) • A String expression that indicates the path to a CSV file to be loaded or the content itself (If the expression points to a file, the file's content is loaded) (similar with Import, Options parameter specifies different options to be used when loading data using the Import method as explained bellow.) • a Safe Array one-dimension or two dimensional to be loaded. If the Source parameter points to a one-dimension safe array, it indicates the rows to be loaded. If the Source parameter refers a two-dimension safe array, the first dimension indicates the rows, while the second indicates the column. If the Options parameter includes the word "reverse", the first dimension indicates the columns, while the second indicates the rows of data (similar with Import, Options parameter specifies different options to be used when loading data using the Import method as explained bellow.)
Options as Variant	A String expression that specifies different options to be used when loading data using the Import method as explained bellow (similar with Import).

Return	Description
Variant	Reserved for future use only.

The AppendData(Source, Options, Result) method appends data to the control. the AppendData method does not clear the data already loaded into the control. The

AppendData method can add a new data source (ADO or DAO recordset), XML file (previously saved by SaveXML method) or anything that Import method accept. The ClearData method clears the control's data. You can use the Layout property to store the control's layout and to restore the layout later.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method
- The user can drag and drop any TXT or XML files to the control. Use the AllowDrop property on False, to prevent loading the data-files (TXT, XML files), by drag and drop, into the control.

method Pivot.AttachTemplate (Template as Variant)

Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.

Type	Description
Template as Variant	A string expression that specifies the Template to execute.

The AttachTemplate/x-script code is a simple way of calling control/object's properties, methods/events using strings. The AttachTemplate features allows you to attach a x-script code to the component. The AttachTemplate method executes x-script code (including events), from a string, file or a safe array of bytes. This feature allows you to run any x-script code for any configuration of the component /COM, /NET or /WPF. Exontrol owns the x-script implementation in its easiest form and it does not require any VB engine or whatever to get executed. The x-script code can be converted to several programming languages using the eXHelper tool.

The following sample opens the Windows Internet Explorer once the user clicks the control (/COM version):

```
AttachTemplate("handle Click(){ CreateObject(`internetexplorer.application`){ Visible =
True; Navigate(`https://www.exontrol.com`) } } ")
```

This script is equivalent with the following VB code:

```
Private Sub Pivot1_Click()
   With CreateObject("internetexplorer.application")
      .Visible = True
      .Navigate ("https://www.exontrol.com")
   End With
End Sub
```

The AttachTemplate/x-script syntax in BNF notation is defined like follows:

```
<x-script> := <lines>
<lines> := <line>[<eol> <lines>] | <block>
<block> := <call> [<eol>] { [<eol>] <lines> [<eol>] } [<eol>]
<eol> := ";" | "\r\n"
<line> := <dim> | <createobject> | <call> | <set> | <comment> | <handle>[<eol>]{[<eol>]
<lines>[<eol>]}[<eol>]
<dim> := "DIM" <variables>
<variables> := <variable> [, <variables>]
```

```
<variable> := "ME" | <identifier>
<createobject> := "CREATEOBJECT(`"<type>"`)"
<call> := <variable> | <property> | <variable>"."<property> | <createobject>"."<property>
<property> := [<property>"."]<identifier>["("<parameters>")"]
<set> := <call> "=" <value>
<property> := <identifier> | <identifier>"("[<parameters>]")"
<parameters> := <value> [","<parameters>]
<value> := <boolean> | <number> | <color> | <date> | <string> | <createobject> | <call>
<boolean> := "TRUE" | "FALSE"
<number> := "0X"<hexa> | ["-"]<integer>["."<integer>]
<digit10> := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<digit16> := <digit10> | A | B | C | D | E | F
<integer> := <digit10>[<integer>]
<hexa> := <digit16>[<hexa>]
<color> := "RGB("<integer>","<integer>","<integer>")"
<date> := "#"<integer>"/"<integer>"/"<integer>" "[<integer>":"<integer>":"<integer>"]"#"
<string> := ""<text>"" | "`"<text>"`"
<comment> := "'"<text>
<handle> := "handle " <event>
<event> := <identifier>"("[<eparameters>]")"
<eparameters> := <eparameter> [","<eparameters>]
<parameters> := <identifier>
```

where:

<identifier> indicates an identifier of the variable, property, method or event, and should start with a letter.
<type> indicates the type the CreateObject function creates, as a progID for /COM version or the assembly-qualified name of the type to create for /NET or /WPF version
<text> any string of characters

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character.

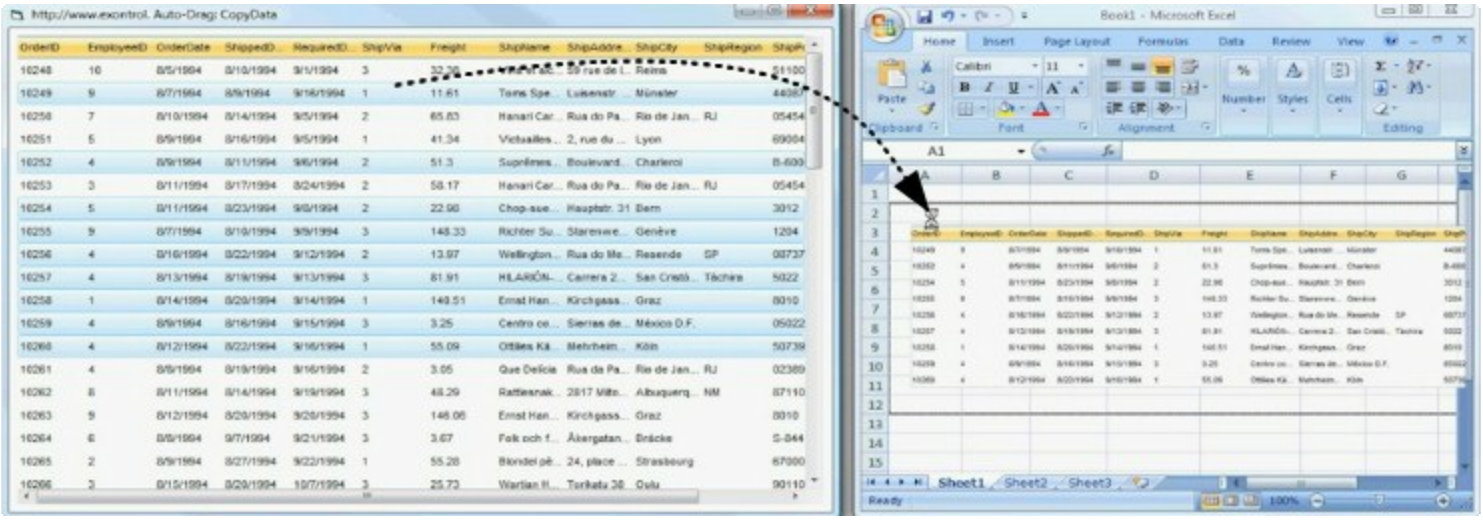
The advantage of the AttachTemplate relative to Template / ExecuteTemplate is that the AttachTemplate can add handlers to the control events.

property Pivot.AutoDrag as AutoDragEnum

Gets or sets a value that indicates the way the component supports the AutoDrag feature.

Type	Description
AutoDragEnum	An AutoDragEnum expression that specifies what the control does once the user clicks and start dragging an item.

By default, the AutoDrag property is exAutoDragNone(0). The AutoDrag feature indicates what the control does when the user clicks an item and starts dragging it. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on. The SingleSel property specifies whether the control supports single or multiple selection. The AutoDrag feature adds automatically Drag and Drop.



The drag and drop operation starts:

- once the user clicks and moves the cursor up or down, if the SingleSel property is True.
- once the user clicks, and waits for a short period of time, if SingleSel property is False (multiple items in selection is allowed). In this case, you can drag and drop any item that is not selected, or a contiguously selection

Once the drag and drop operation starts the mouse pointer is changed to MOVE cursor if the operation is possible, else if the Drag and Drop operation fails or if it is not possible, the mouse pointer is changed to NO cursor.

Use the AutoDrag property to allow Drag and Drop operations like follows:

- Ability to drag and drop the data as *text*, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant

- Ability to drag and drop the data as it *looks*, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant
- Ability to smoothly scroll the control's content moving the mouse cursor up or down
- and more ...

property Pivot.BackColor as Color

Specifies the control's background color.

Type	Description
Color	A Color expression that specifies the control's background color.

The BackColor property specifies the control's background color. The ForeColor property specifies the control's foreground color. The BackColor property changes the background color of the control's pivot bar. You can use the Background(exPivotBarBackColor) property to specify a different background color for the control's pivot bar. Use the Appearance property to specify the visual appearance of the control's frame. The BackColorAlternate property specifies a different background color for even/odd rows.

property Pivot.BackColorAlternate as Color

Specifies the background color used to display alternate items in the control.

Type	Description
Color	A color expression that indicates the alternate background color.

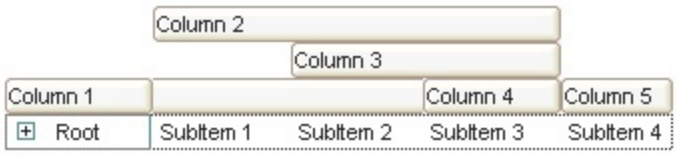
By default, the control's BackColorAlternate property is zero. The control ignores the BackColorAlternate property if it is 0 (zero). Use the BackColor property to specify the control's background color. Use the SelBackColor property to specify the selection background color.

property Pivot.BackColorHeader as Color

Specifies the header's background color.

Type	Description
Color	A color expression that indicates the background color for the control's header.

Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the BackColor property to specify the control's background color.



The following VB sample changes the visual appearance for the control's header. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the BackColorHeader property to indicates the index of the skin that we want to use. The sample applies the " " to the control' header bar:

```
With Pivot1
   With .VisualAppearance
      .Add &H24, App.Path + "\header.ebn"
   End With
   .BackColorHeader = &H24000000
End With
```

The following C++ sample changes the visual aspect of the control' header bar:

```
#include "Appearance.h"
m_pivot.GetVisualAppearance().Add( 0x24,
COleVariant(_T("D:\\Temp\\ExPivot.Help\\header.ebn")) );
m_pivot.SetBackColorHeader( 0x24000000 );
```

The following VB.NET sample changes the visual aspect of the control' header bar:

```
With AxPivot1
   With .VisualAppearance
      .Add(&H24, "D:\Temp\ExPivot.Help\header.ebn")
   End With
   .Template = "BackColorHeader = 603979776"
```

```
End With
```

The 603979776 value indicates the &H24000000 in hexadecimal.

The following C# sample changes the visual aspect of the control' header bar:

```
axPivot1.VisualAppearance.Add(0x24, "D:\\Temp\\ExPivot.Help\\header.ebn");
axPivot1.Template = "BackColorHeader = 603979776";
```

The 603979776 value indicates the 0x24000000 in hexadecimal.

The following VFP sample changes the visual aspect of the control' header bar:

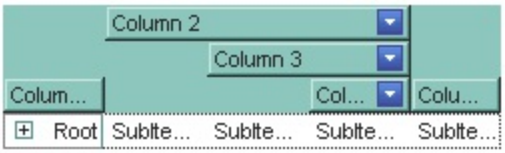
```
With thisform.Pivot1
   With .VisualAppearance
      .Add(36, "D:\Temp\ExPivot.Help\header.ebn")
   EndWith
   .BackColorHeader = 603979776
EndWith
```

property Pivot.Background(Part as BackgroundPartEnum) as Color

Returns or sets a value that indicates the background color for parts in the control.

Type	Description
Part as BackgroundPartEnum	A BackgroundPartEnum expression that indicates a part in the control.
Color	A Color expression that indicates the background color for a specified part. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The Background property specifies a background color or a visual appearance for specific parts in the control. If the Background property is 0, the control draws the part as default. Use the Add method to add new skins to the control. Use the Remove method to remove a specific skin from the control. Use the Clear method to remove all skins in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while init the control. Use the Refresh method to refresh the control.



The following VB sample changes the visual appearance for the "drop down" filter button. The sample applies the skin "" to the "drop down" filter buttons:

```
With Pivot1
   With .VisualAppearance
      .Add &H1, App.Path + "\fbardd.ebn"
   End With
   .Background(exHeaderFilterBarButton) = &H1000000
End With
```

The following C++ sample changes the visual appearance for the "drop down" filter button:

```
#include "Appearance.h"
m_pivot.GetVisualAppearance().Add( 0x01,
COleVariant(_T("D:\\Temp\\ExPivot.Help\\fbardd.ebn")) );
```

```
m_pivot.SetBackground( 0 /*exHeaderFilterBarButton*/, 0x1000000 );
```

The following VB.NET sample changes the visual appearance for the "drop down" filter button:

```
With AxPivot1
   With .VisualAppearance
      .Add(&H1, "D:\Temp\ExPivot.Help\fbardd.ebn")
   End With
   .set_Background(EXPIVOTLib.BackgroundPartEnum.exHeaderFilterBarButton,
&H1000000)
End With
```

The following C# sample changes the visual appearance for the "drop down" filter button:

```
axPivot1.VisualAppearance.Add(0x1, "D:\\Temp\\ExPivot.Help\\fbardd.ebn");
axPivot1.set_Background(EXPIVOTLib.BackgroundPartEnum.exHeaderFilterBarButton,
0x1000000);
```

The following VFP sample changes the visual appearance for the "drop down" filter button:

```
With thisform.Pivot1
   With .VisualAppearance
      .Add(1, "D:\Temp\ExPivot.Help\fbardd.ebn")
   EndWith
   .Object.Background(0) = 16777216
EndWith
```

The 16777216 value is the 0x1000000 value in hexadecimal.

method Pivot.BeginUpdate ()

Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.

Type	Description

This method prevents the control from painting until the EndUpdate method is called. The BeginUpdate and EndUpdate methods increases the speed of loading your events, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too. You can use the Refresh method to refresh the control's content.

property Pivot.BorderHeight as Long

Sets or retrieves a value that indicates the border height of the control.

Type	Description
Long	A long expression that specifies the height of the border being applied to the top and bottom side of the control.

By default, the BorderHeight property is 0. The BorderHeight property specifies the height of the border in pixels, being applied to the top and bottom side of the control. The BorderWidth property specifies the width of the border on the left and right side of the control. The borders delimit the margin of the control and the client area, where the calendar and the pivot panel is displayed.

property Pivot.BorderWidth as Long

Sets or retrieves a value that indicates the border width of the control.

Type	Description
Long	A long expression that specifies the width of the border being applied to the left and right side of the control.

By default, the BorderWidth property is 0. The BorderWidth property specifies the width of the border in pixels, being applied to the left and right side of the control. The BorderHeight property specifies the height of the border on the top and bottom side of the control. The borders delimit the margin of the control and the client area, where the calendar and the pivot panel is displayed.

property Pivot.CheckImage(State as CheckStateEnum) as Long

Retrieves or sets a value that indicates the image used by cells of checkbox type.

Type	Description
State as CheckStateEnum	A CheckStateEnum expression that indicates the check's state: 0 means unchecked, 1 means checked, and 2 means partial checked.
Long	A long expression that indicates the index of image used to paint the cells of check box types. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part.

Use CheckImage and RadioImage properties to define icons for radio and check box cells. The CheckImage property defines the index of the icon being used by check boxes. The ImageSize property defines the size (width/height) of the check-boxes.

method Pivot.ClearData ()

Removes the control's data.

Type	Description

The ClearData method clears the control's data. The ClearData method clears/empties the DataColumns collection and the PivotRows, PivotColumns, PivotTotals properties.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method

The following properties may be used to group and summarize the data, once it is loaded:

- PivotRows property specifies the list of DATA columns that determines the first column in the control's list. In other words, the Group-By columns
- PivotColumns property specifies the list of DATA columns that determines the rest of the columns to be displayed on the control's list
- PivotTotals property specifies the list of total/sub-total functions to be displayed on the control's list.

method Pivot.ClearFilter ()

Clears the filter.

Type	Description

method Pivot.CollapseAll ()

Collapses all rows.

Type	Description

property Pivot.ColumnAutoResize as Boolean

Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.

Type	Description
Boolean	A boolean expression indicating whether the control will automatically size its visible columns to fit on the control's client width.

By default, the ColumnAutoResize property is False. Use the ColumnAutoResize property to fit all your columns in the client area. Use the DefaultColumnWidth property to specify the default column's width.

property Pivot.ColumnFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Variant

Retrieves the column from the point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Variant	A NULL expression (VT_NULL) if no column at the point, or the column's name from the point.

The ColumnFromPoint property gets the column's name from the cursor. The ValueFromPoint property gets the value from the cursor, while it hovers the control's list part. The ColumnFromPoint(-1,-1) gets the column from the current cursor position, while it hovers the control's list part. The DataColumnFromPoint property retrieves the index of the data column from the point, or -1 if not data column is found.

method Pivot.Copy ()

Copies the control's content to the clipboard, in the EMF format.

Type	Description

Use the Copy method to copy the control's content to the clipboard, in Enhanced Metafile (EMF) format. The Enhanced Metafile format is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following:

Built-in scaling information
Built-in descriptions that are saved with the file
Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The Copy method copies the control's header if it's visible, and all visible items. Use the CopyTo method to copy the control's view to an EMF file. The items are not expanded, they are listed in the order as they are displayed on the screen. Use the HeaderVisible property to show or hide the control's header. The background of the copied control is transparent. You can use the Export method to export the control's DATA in CSV format.

The following VB sample saves the control's content to a EMF file, when user presses the CTRL+C key:

```
Private Sub Pivot1_KeyDown(KeyCode As Integer, Shift As Integer)
   If (KeyCode = vbKeyC) And Shift = 2 Then
      Clipboard.Clear
      Pivot1.Copy
      SavePicture Clipboard.GetData(), App.Path & "\test.emf"
   End If
End Sub
```

Now, you can open your MS Windows Word application, and you can insert the file using the Insert\Picture\From File menu, or by pressing the CTRL+V key to paste the clipboard.

The following C++ function saves the clipboard's data (EMF format) to a picture file:

```
BOOL saveEMFtoFile( LPCTSTR szFileName )
```

```
{
   BOOL bResult = FALSE;
   if ( ::OpenClipboard( NULL ) )
   {
      CComPtr spPicture;
      PICTDESC pictDesc = {0};
      pictDesc.cbSizeofstruct = sizeof(pictDesc);
      pictDesc.emf.hemf = (HENHMETAFILE)GetClipboardData( CF_ENHMETAFILE );
      pictDesc.picType = PICTYPE_ENHMETAFILE;
      if ( SUCCEEDED( OleCreatePictureIndirect( &pictDesc;, IID_IPicture, FALSE,
(LPVOID*)&spPicture; ) ) )
      {
         HGLOBAL hGlobal = NULL;
         CComPtr spStream;
         if ( SUCCEEDED( CreateStreamOnHGlobal( hGlobal = GlobalAlloc( GPTR, 0 ), TRUE,
&spStream; ) ) )
         {
            long dwSize = NULL;
            if ( SUCCEEDED( spPicture->SaveAsFile( spStream, TRUE, &dwSize; ) ) )
            {
               USES_CONVERSION;
               HANDLE hFile = CreateFile( szFileName, GENERIC_WRITE, NULL, NULL,
CREATE_ALWAYS, NULL, NULL );
               if ( hFile != INVALID_HANDLE_VALUE )
               {
                  LARGE_INTEGER l = {NULL};
                  spStream->Seek(l, STREAM_SEEK_SET, NULL);
                  long dwWritten = NULL;
                  while ( dwWritten < dwSize )
                  {
                     unsigned long dwRead = NULL;
                     BYTE b[10240] = {0};
                     spStream->Read( &b;, 10240, &dwRead; );
                     DWORD dwBWritten = NULL;
                     WriteFile( hFile, b, dwRead, &dwBWritten;, NULL );
                     dwWritten += dwBWritten;
                  }
```

```
                CloseHandle( hFile );
                bResult = TRUE;
            }
        }
    }
  }
  CloseClipboard();
 }
 return bResult;
}
```

The following VB.NET sample copies the control's content to the clipboard (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear()
With AxPivot1
  .Copy()
End With
```

The following C# sample copies the control's content to a file (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear;
axPivot1.Copy();
```

property Pivot.CopyTo (File as String) as Variant

Exports the control's view to an EMF file.

Type	Description
File as String	A String expression that indicates the name of the file to be saved. If present, the CopyTo property retrieves True, if the operation succeeded, else False it is failed. If the File parameter is missing or empty, the CopyTo property retrieves an one dimension safe array of bytes that contains the EMF content. If the File parameter is not empty, the extension (characters after last dot) determines the graphical/ format of the file to be saved as follows: • ***.bmp *.dib *.rle**, saves the control's content in **BMP** format. • ***.jpg *.jpe *.jpeg *.jfif**, saves the control's content in **JPEG** format. • ***.gif**, , saves the control's content in **GIF** format. • ***.tif *.tiff**, saves the control's content in **TIFF** format. • ***.png**, saves the control's content in **PNG** format. • ***.pdf**, saves the control's content to PDF format. The File argument may carry up to 4 parameters separated by the | character in the following order: ***filename.pdf | paper size | margins | options***. In other words, you can specify the file name of the PDF document, the paper size, the margins and options to build the PDF document. By default, the paper size is 210 **mm** × 297 **mm** (A4 format) and the margins are 12.7 **mm** 12.7 **mm** 12.7 **mm** 12.7 **mm**. The units for the paper size and margins can be **pt** for PostScript Points, **mm** for Millimeters, **cm** for Centimeters, **in** for Inches and **px** for pixels. If PostScript Points are used if unit is missing. For instance, 8.27 in x 11.69 in, indicates the size of the paper in inches. Currently, the options can be **single**, which indicates that the control's content is exported to a single PDF page. For instance, the CopyTo("shot.pdf|33.11 in x 46.81 in|0 0 0 0|single") exports the control's content to an A0 single PDF page, with no margins. • ***.emf** or any other extension determines the control to

	save the control's content in **EMF** format. For instance, the CopyTo("c:\temp\snapshot.png") property saves the control's content in PNG format to snapshot.png file.
Variant	A boolean expression that indicates whether the File was successful saved, or a one dimension safe array of bytes, if the File parameter is empty string.

The CopyTo method copies/exports the control's view to BMP, PNG, JPG, GIF, TIFF, PDF or EMF graphical files, including no scroll bars. Use the Copy method to copy the control's content to the clipboard. You can use the Export method to export the control's DATA in CSV format.

- The **BMP** file format, also known as bitmap image file or device independent bitmap (DIB) file format or simply a bitmap, is a raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter)
- The **JPEG** file format (seen most often with the .jpg extension) is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography.
- The **GIF** (Graphics Interchange Format) is a bitmap image format that was introduced by CompuServe in 1987 and has since come into widespread usage on the World Wide Web due to its wide support and portability.
- The **TIFF** (Tagged Image File Format) is a computer file format for storing raster graphics images, popular among graphic artists, the publishing industry, and both amateur and professional photographers in general.
- The **PNG** (Portable Network Graphics) is a raster graphics file format that supports lossless data compression. PNG was created as an improved, non-patented replacement for Graphics Interchange Format (GIF), and is the most used lossless image compression format on the Internet
- The **PDF** (Portable Document Format) is a file format used to present documents in a manner independent of application software, hardware, and operating systems. Each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, fonts, graphics, and other information needed to display it.
- The EMF (Enhanced Metafile Format) is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following

  Built-in scaling information
  Built-in descriptions that are saved with the file
  Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The following VB sample saves the control's content to a file:

```
If (Pivot1.CopyTo("c:\temp\test.emf")) Then
   MsgBox "test.emf file created, open it using the mspaint editor."
End If
```

The following VB sample prints the EMF content (as bytes, File parameter is empty string):

```
Dim i As Variant
For Each i In Pivot1.CopyTo("")
   Debug.Print i
Next
```

property Pivot.DataColumnFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

Retrieves the index of the data column from the point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Long	A Long expression that determines the index of the data column from the cursor. The DataColumnFromPoint property returns -1, if no data column is found at the cursor.

The DataColumnFromPoint property retrieves the index of the data column from the point, or -1 if not data column is found. The ColumnFromPoint(-1,-1) gets the column from the current cursor position, while it hovers the control's list part. The ValueFromPoint property gets the value from the cursor, while it hovers the control's list part.

property Pivot.DataColumns as Columns

Retrieves the Data Columns collection of the pivot control.

Type	Description
Columns	A Columns object that holds a collection of Column object.

By default, the DataColumns collection is empty. Once you load data to the controls, the DataColumns collection is filled. The PivotColumnsFloatBarVisible property specifies whether the Columns collection is displayed to a floating bar, so user can drag and drop columns to the control's pivot bar so it gets data summarized. The ClearData method clears the control's data.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method

You can use the DataColumns property to rename the columns, change the column's alignment, specify the sorting-filtering type, and so on.

property Pivot.DataSource as Object

Retrieves or sets a value that indicates the data source for object.

Type	Description
Object	An Object that defines the control's data. Currently, the control accepts ADO.Recordset, ADODB.Recordset objects, DAO recordset.

The DataSource property may be used to load data from a table or a record set using the ADO or DAO recordsets. Microsoft's ActiveX Data Objects (ADO) is a set of Component Object Model (COM) objects for accessing data sources. In computer software, a data access object (DAO) is an object that provides an abstract interface to some type of database or other persistence mechanism. By mapping application calls to the persistence layer, DAOs provide some specific data operations without exposing details of the database. The DataColumns property accesses the control's Columns collection, so you can rename or specify the column's type once the control's data is loaded. The PivotColumnsFloatBarVisible property specifies whether the Columns collection is displayed to a floating bar, so user can drag and drop columns to the control's pivot bar so it gets data summarized. The ClearData method clears the control's data. Use the DisplayPivotData property to specify the number of rows to be displayed on the control's list. The LoadHeadersOnly property loads the headers only, so no data is loaded.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method
- The user can drag and drop any TXT or XML files to the control.

The AppendData method appends data to the control (prevents clearing data already loaded).

The following properties may be used to group and summarize the data, once it is loaded:

- PivotRows property specifies the list of DATA columns that determines the first column in the control's list. In other words, the Group-By columns
- PivotColumns property specifies the list of DATA columns that determines the rest of the columns to be displayed on the control's list
- PivotTotals property specifies the list of total/sub-total functions to be displayed on the control's list.

Is it possible to load data from a data source?

VBA (MS Access, ...)

```
With Pivot1
   .DataSource = CurrentDb.OpenRecordset("Data")
End With
```

VB6

```
With Pivot1
   Set rs = CreateObject("ADOR.Recordset")
   With rs
      .Open "Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb",3,3
   End With
   .DataSource = rs
End With
```

VB.NET

```
Dim rs
With Expivot1
   rs = New ADODB.Recordset()
   With rs
      .Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb",3,3)
   End With
   .DataSource = rs
End With
```

VB.NET for /COM

```
Dim rs
With AxPivot1
   rs = CreateObject("ADOR.Recordset")
   With rs
      .Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb",3,3)
   End With
   .DataSource = rs
```

```
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

   #import <ExPivot.dll>
   using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
/*
   Includes the definition for CreateObject function like follows:

   #include <comdef.h>
   IUnknownPtr CreateObject( BSTR Object )
   {
      IUnknownPtr spResult;
      spResult.CreateInstance( Object );
      return spResult;
   };

*/
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'ADODB' for the library: 'Microsoft ActiveX Data Objects
6.0 Library'

   #import <msado15.dll> rename("EOF","REOF")
*/
ADODB::_RecordsetPtr rs = ::CreateObject(L"ADOR.Recordset");
   rs->Open("Data",_bstr_t("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\Access2007\\sample.accdb",ADODB::adOpenStatic,

spPivot1->PutDataSource(((ADODB::_RecordsetPtr)(rs)));
```

C++ Builder

```
/*
   Select the Component\Import Component...\Import a Type Library,
   to import the following Type Library:

      Microsoft ActiveX Data Objects 6.0 Library

   TypeLib: C:\Program Files\Common Files\System\ado\msado15.dll

   to define the namespace: Adodb_tlb
*/
//#include "ADODB_TLB.h"
Adodb_tlb::_RecordsetPtr rs = Variant::CreateObject(L"ADOR.Recordset");
   rs-
>Open(TVariant("Data"),TVariant(String("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\Access2007\\sample.accdb"),Adodb_tlb::CursorType

Pivot1->DataSource = (IDispatch*)rs;
```

C#

```
// Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.
ADODB.Recordset rs = new ADODB.Recordset();
   rs.Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\Access2007\\sample.accdb",exontrol.ADODB.Curso

expivot1.DataSource = (rs as ADODB.Recordset);
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
   var rs = new ActiveXObject("ADOR.Recordset");
      rs.Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\Access2007\\sample.accdb",3,3,null);
   Pivot1.DataSource = rs;
</SCRIPT>
```

C# for /COM

```
// Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.
ADODB.Recordset rs = new ADODB.Recordset();
   rs.Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\Access2007\\sample.accdb",ADODB.CursorTypeEnu

axPivot1.DataSource = (rs as ADODB.Recordset);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   anytype rs;
   str var_s;
   ;

   super();

   // Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your
project.
   rs = COM::createFromObject(new ADODB.Recordset()); rs = rs;
      var_s = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\Access2007\\sample.accdb";

rs.Open("Data",COMVariant::createFromStr(var_s),3/*adOpenStatic*/,3/*adLockOptimis

   expivot1.DataSource(rs);
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   rs := (ComObj.CreateComObject(ComObj.ProgIDToClassID('ADOR.Recordset')) as
ADODB.Recordset);
   with rs do
   begin
      Open('Data','Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb',3,3,Nil);
   end;
   DataSource := (rs as ADODB.Recordset);
end
```

Delphi (standard)

```
with Pivot1 do
begin
   rs :=
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('ADOR.Recordset')))
as ADODB_TLB.Recordset);
   with rs do
   begin
      Open('Data','Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb',3,3,Null);
   end;
   DataSource := (IUnknown(rs) as ADODB_TLB.Recordset);
end
```

VFP

```
with thisform.Pivot1
   rs = CreateObject("ADOR.Recordset")
   with rs
      var_s = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb"
      .Open("Data",var_s,3,3)
   endwith
```

```
   .DataSource = rs
endwith
```

dBASE Plus

```
local oPivot,rs

oPivot = form.Activex1.nativeObject
rs = new OleAutoClient("ADOR.Recordset")
   rs.Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb",3,3)
oPivot.DataSource = rs
```

XBasic (Alpha Five)

```
Dim oPivot as P
Dim rs as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
rs = OLE.Create("ADOR.Recordset")
   rs.Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb",3,3)
oPivot.DataSource = rs
```

Visual Objects

```
local rs as _Recordset

// Generate Source for 'Microsoft ActiveX Data Objects 6.0 Library' server
from Tools\Automation Server...
rs := _Recordset{"ADOR.Recordset"}
   rs:Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb",3,3,0)
oDCOCX_Exontrol1:DataSource := _Recordset{rs}
```

PowerBuilder

```
OleObject oPivot,rs

oPivot = ole_1.Object
rs = CREATE OLEObject
rs.ConnectToNewObject("ADOR.Recordset")
   rs.Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb",3,3)
oPivot.DataSource = rs
```

property Pivot.DefaultColumnWidth as Long

Retrieves or sets a value that indicates the default column width.

Type	Description
Long	A Long expression that defines the default width of the column.

By default, the DefaultColumnWidth property is 128 pixels. Use the DefaultColumnWidth property to specify the default for the width of the columns. The DefaultItemHeight property specifies the default height of the rows/items. The HeaderHeight property specifies the height of the control's header, and so the height of the columns to be displayed on the control's pivot bar. Use the Refresh method to refresh the control.

property Pivot.DefaultItemHeight as Long

Retrieves or sets a value that indicates the default item height.

Type	Description
Long	A long expression indicates the default item height.

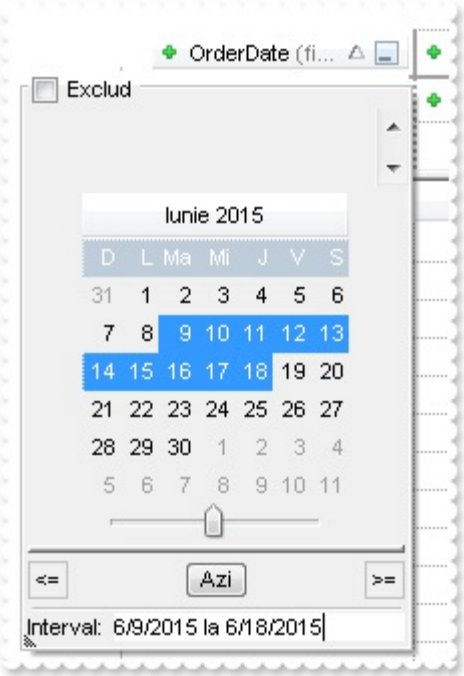
By default, the DefaultColumnWidth property is 18 pixels. The DefaultItemHeight property specifies the default height of the rows/items. Use the DefaultColumnWidth property to specify the default for the width of the columns. The HeaderHeight property specifies the height of the control's header, and so the height of the columns to be displayed on the control's pivot bar. Use the Refresh method to refresh the control.

property Pivot.Description(Type as DescriptionTypeEnum) as String

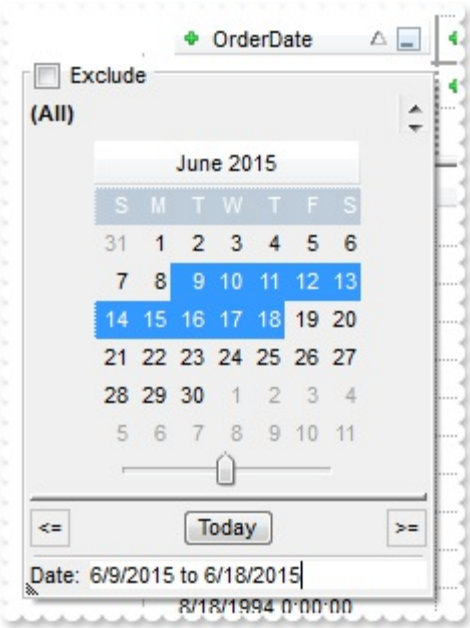
Changes descriptions for control objects.

Type	Description
Type as DescriptionTypeEnum	A DescriptionTypeEnum expression that indicates the caption begin changed
String	A String expression that defines the new caption to be shown.

The Description property defines pre-defined captions being shown on the control.



instead of



The following samples show how you can change the name for months in the drop-down filter window (localization)?

VBA (MS Access, Excell...)

```
With Pivot1
   Debug.Print( .Description(17) )
   .Description(17) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie"
   .Description(0) = "(Toate)"
   .Description(1) = "(Gol)"
   .Description(2) = "(Plin)"
   .Description(3) = "Filtreaza:"
   .Description(16) = "Azi"
```

```
    .Description(18) = "D L Ma Mi J V S"
    .Description(25) = "Exclud"
    .Description(26) = "Coloane"
    .Description(11) = "si"
    .Description(12) = "Data:"
    .Description(15) = "Data"
    .Description(13) = "la"
    .Description(24) = "nu"
    .Description(23) = "sau"
    .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
    .DataColumns.Item("OrderDate").SortType = 2
    .PivotRows = "9"
    .Refresh
End With
```

VB6

```
With Pivot1
    Debug.Print( .Description(exFilterBarDateMonths) )
    .Description(exFilterBarDateMonths) = "Ianuarie Februarie Martie Aprilie Mai Iunie
Iulie August Septembrie Octombrie Noiembrie Decembrie"
    .Description(exFilterBarAll) = "(Toate)"
    .Description(exFilterBarBlanks) = "(Gol)"
    .Description(exFilterBarNonBlanks) = "(Plin)"
    .Description(exFilterBarFilterForCaption) = "Filtreaza:"
    .Description(exFilterBarDateTodayCaption) = "Azi"
    .Description(exFilterBarDateWeekDays) = "D L Ma Mi J V S"
    .Description(exFilterBarExclude) = "Exclud"
    .Description(exColumnsFloatBar) = "Coloane"
    .Description(exFilterBarAnd) = "si"
    .Description(exFilterBarDate) = "Data:"
    .Description(exFilterBarDateTitle) = "Data"
    .Description(exFilterBarDateTo) = "la"
    .Description(exFilterBarNot) = "nu"
    .Description(exFilterBarOr) = "sau"
    .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
    .DataColumns.Item("OrderDate").SortType = SortDate
```

```
    .PivotRows = "9"
    .Refresh
End With
```

VB.NET

```
With Expivot1
    Debug.Print(
.get_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths)
)

.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths,"Ia
 Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie
Decembrie")
    .set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarAll,"
(Toate)")
    .set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarBlanks,"
(Gol)")

.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarNonBlanks,"
(Plin)")

.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarFilterForCaptio


.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTodayCapt


.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateWeekDays
 L Ma Mi J V S")

.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarExclude,"Exclud


.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exColumnsFloatBar,"Coloa

    .set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarAnd,"si")
```

```
.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDate,"Data:")

.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTitle,"Data

    .set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTo,"la")
    .set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarNot,"nu")
    .set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarOr,"sau")
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .DataColumns.Item("OrderDate").SortType =
exontrol.EXPIVOTLib.SortTypeEnum.SortDate
    .PivotRows = "9"
    .Refresh()
End With
```

VB.NET for /COM

```
With AxPivot1
    Debug.Print(
.get_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths) )

.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths,"Ianuarie
Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie
Decembrie")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarAll,"(Toate)")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarBlanks,"(Gol)")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarNonBlanks,"(Plin)")

.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarFilterForCaption,"Filtrea



.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTodayCaption,"Azi"

    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateWeekDays,"D L
Ma Mi J V S")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarExclude,"Exclud")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exColumnsFloatBar,"Coloane")
```

```
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarAnd,"si")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDate,"Data:")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTitle,"Data")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTo,"la")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarNot,"nu")
    .set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarOr,"sau")
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .DataColumns.Item("OrderDate").SortType = EXPIVOTLib.SortTypeEnum.SortDate
    .PivotRows = "9"
    .Refresh()
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

    #import <ExPivot.dll>
    using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
OutputDebugStringW( spPivot1-
>GetDescription(EXPIVOTLib::exFilterBarDateMonths) );
spPivot1->PutDescription(EXPIVOTLib::exFilterBarDateMonths,L"Ianuarie Februarie
Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarAll,L"(Toate)");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarBlanks,L"(Gol)");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarNonBlanks,L"(Plin)");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarFilterForCaption,L"Filtreaza:");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarDateTodayCaption,L"Azi");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarDateWeekDays,L"D L Ma Mi J V
S");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarExclude,L"Exclud");
spPivot1->PutDescription(EXPIVOTLib::exColumnsFloatBar,L"Coloane");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarAnd,L"si");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarDate,L"Data:");
```

```
spPivot1->PutDescription(EXPIVOTLib::exFilterBarDateTitle,L"Data");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarDateTo,L"la");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarNot,L"nu");
spPivot1->PutDescription(EXPIVOTLib::exFilterBarOr,L"sau");
spPivot1->Import("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt",vtMissing);
spPivot1->GetDataColumns()->GetItem("OrderDate")-
>PutSortType(EXPIVOTLib::SortDate);
spPivot1->PutPivotRows(L"9");
spPivot1->Refresh();
```

C++ Builder

```
OutputDebugString( Pivot1-
>Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarDateMonths] );
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarDateMonths] =
L"Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie
Noiembrie Decembrie";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarAll] = L"
(Toate)";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarBlanks] = L"
(Gol)";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarNonBlanks] =
L"(Plin)";
Pivot1-
>Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarFilterForCaption] =
L"Filtreaza:";
Pivot1-
>Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarDateTodayCaption] =
L"Azi";
Pivot1-
>Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarDateWeekDays] = L"D
L Ma Mi J V S";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarExclude] =
L"Exclud";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exColumnsFloatBar] =
```

```
L"Coloane";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarAnd] = L"si";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarDate] =
L"Data:";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarDateTitle] =
L"Data";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarDateTo] = L"la";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarNot] = L"nu";
Pivot1->Description[Expivotlib_tlb::DescriptionTypeEnum::exFilterBarOr] = L"sau";
Pivot1->Import(TVariant("C:\\Program
Files\\Exontrol\\ExPivot\\Sample\\data.txt"),TNoParam());
Pivot1->DataColumns->get_Item(TVariant("OrderDate"))->SortType =
Expivotlib_tlb::SortTypeEnum::SortDate;
Pivot1->PivotRows = L"9";
Pivot1->Refresh();
```

C#

```
System.Diagnostics.Debug.Print(
expivot1.get_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateM
 );
expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateM
 Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie
Decembrie");
expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarAll,"
(Toate)");
expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarBlanks
(Gol)");
expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarNonBl
(Plin)");
expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarFilterF

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTo

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateW
 L Ma Mi J V S");
```

```
expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarExclud

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exColumnsFloatE

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarAnd,"s

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDate,"

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTi

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTo

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarNot,"n

expivot1.set_Description(exontrol.EXPIVOTLib.DescriptionTypeEnum.exFilterBarOr,"sau

expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
expivot1.DataColumns["OrderDate"].SortType =
exontrol.EXPIVOTLib.SortTypeEnum.SortDate;
expivot1.PivotRows = "9";
expivot1.Refresh();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686"
id="Pivot1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   alert( Pivot1.Description(17) );
   Pivot1.Description(17) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August
Septembrie Octombrie Noiembrie Decembrie";
   Pivot1.Description(0) = "(Toate)";
   Pivot1.Description(1) = "(Gol)";
```

```
   Pivot1.Description(2) = "(Plin)";
   Pivot1.Description(3) = "Filtreaza:";
   Pivot1.Description(16) = "Azi";
   Pivot1.Description(18) = "D L Ma Mi J V S";
   Pivot1.Description(25) = "Exclud";
   Pivot1.Description(26) = "Coloane";
   Pivot1.Description(11) = "si";
   Pivot1.Description(12) = "Data:";
   Pivot1.Description(15) = "Data";
   Pivot1.Description(13) = "la";
   Pivot1.Description(24) = "nu";
   Pivot1.Description(23) = "sau";
   Pivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
   Pivot1.DataColumns.Item("OrderDate").SortType = 2;
   Pivot1.PivotRows = "9";
   Pivot1.Refresh();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686"
id="Pivot1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With Pivot1
      alert( .Description(17) )
      .Description(17) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August
Septembrie Octombrie Noiembrie Decembrie"
      .Description(0) = "(Toate)"
      .Description(1) = "(Gol)"
      .Description(2) = "(Plin)"
      .Description(3) = "Filtreaza:"
```

```
        .Description(16) = "Azi"
        .Description(18) = "D L Ma Mi J V S"
        .Description(25) = "Exclud"
        .Description(26) = "Coloane"
        .Description(11) = "si"
        .Description(12) = "Data:"
        .Description(15) = "Data"
        .Description(13) = "la"
        .Description(24) = "nu"
        .Description(23) = "sau"
        .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
        .DataColumns.Item("OrderDate").SortType = 2
        .PivotRows = "9"
        .Refresh
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
System.Diagnostics.Debug.Print(
axPivot1.get_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths)
);
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths,"Ia
 Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie
Decembrie");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarAll,"(Toate)");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarBlanks,"
(Gol)");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarNonBlanks,"
(Plin)");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarFilterForCaptio

axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTodayCapt
```

```
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateWeekDays,
 L Ma Mi J V S");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarExclude,"Exclud

axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exColumnsFloatBar,"Coloa

axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarAnd,"si");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDate,"Data:");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTitle,"Data

axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTo,"la");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarNot,"nu");
axPivot1.set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarOr,"sau");
axPivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt",null);
axPivot1.DataColumns["OrderDate"].SortType =
EXPIVOTLib.SortTypeEnum.SortDate;
axPivot1.PivotRows = "9";
axPivot1.Refresh();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Column;
   anytype var_Column;
   ;

   super();

   print( expivot1.Description(17/*exFilterBarDateMonths*/) );
   expivot1.Description(17/*exFilterBarDateMonths*/,"Ianuarie Februarie Martie
Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie");
   expivot1.Description(0/*exFilterBarAll*/,"(Toate)");
   expivot1.Description(1/*exFilterBarBlanks*/,"(Gol)");
   expivot1.Description(2/*exFilterBarNonBlanks*/,"(Plin)");
   expivot1.Description(3/*exFilterBarFilterForCaption*/,"Filtreaza:");
```

```
   expivot1.Description(16/*exFilterBarDateTodayCaption*/,"Azi");
   expivot1.Description(18/*exFilterBarDateWeekDays*/,"D L Ma Mi J V S");
   expivot1.Description(25/*exFilterBarExclude*/,"Exclud");
   expivot1.Description(26/*exColumnsFloatBar*/,"Coloane");
   expivot1.Description(11/*exFilterBarAnd*/,"si");
   expivot1.Description(12/*exFilterBarDate*/,"Data:");
   expivot1.Description(15/*exFilterBarDateTitle*/,"Data");
   expivot1.Description(13/*exFilterBarDateTo*/,"la");
   expivot1.Description(24/*exFilterBarNot*/,"nu");
   expivot1.Description(23/*exFilterBarOr*/,"sau");
   expivot1.Import("C:\\Program Files\\Exontrol\\ExPivot\\Sample\\data.txt");
   var_Column =
COM::createFromObject(expivot1.DataColumns()).Item("OrderDate"); com_Column =
var_Column;
   com_Column.SortType(2/*SortDate*/);
   expivot1.PivotRows("9");
   expivot1.Refresh();
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   OutputDebugString(
get_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths) );

set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateMonths,'Ianuarie
Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie
Decembrie');
   set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarAll,'(Toate)');
   set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarBlanks,'(Gol)');
   set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarNonBlanks,'(Plin)');

set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarFilterForCaption,'Filtreaz



set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTodayCaption,'Azi');
```

```
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateWeekDays,'D L Ma Mi J V S');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarExclude,'Exclud');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exColumnsFloatBar,'Coloane');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarAnd,'si');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDate,'Data:');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTitle,'Data');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarDateTo,'la');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarNot,'nu');
  set_Description(EXPIVOTLib.DescriptionTypeEnum.exFilterBarOr,'sau');
  Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Nil);
  DataColumns.Item['OrderDate'].SortType := EXPIVOTLib.SortTypeEnum.SortDate;
  PivotRows := '9';
  Refresh();
end
```

Delphi (standard)

```
with Pivot1 do
begin
  OutputDebugString( Description[EXPIVOTLib_TLB.exFilterBarDateMonths] );
  Description[EXPIVOTLib_TLB.exFilterBarDateMonths] := 'Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie';
  Description[EXPIVOTLib_TLB.exFilterBarAll] := '(Toate)';
  Description[EXPIVOTLib_TLB.exFilterBarBlanks] := '(Gol)';
  Description[EXPIVOTLib_TLB.exFilterBarNonBlanks] := '(Plin)';
  Description[EXPIVOTLib_TLB.exFilterBarFilterForCaption] := 'Filtreaza:';
  Description[EXPIVOTLib_TLB.exFilterBarDateTodayCaption] := 'Azi';
  Description[EXPIVOTLib_TLB.exFilterBarDateWeekDays] := 'D L Ma Mi J V S';
  Description[EXPIVOTLib_TLB.exFilterBarExclude] := 'Exclud';
  Description[EXPIVOTLib_TLB.exColumnsFloatBar] := 'Coloane';
  Description[EXPIVOTLib_TLB.exFilterBarAnd] := 'si';
  Description[EXPIVOTLib_TLB.exFilterBarDate] := 'Data:';
  Description[EXPIVOTLib_TLB.exFilterBarDateTitle] := 'Data';
  Description[EXPIVOTLib_TLB.exFilterBarDateTo] := 'la';
  Description[EXPIVOTLib_TLB.exFilterBarNot] := 'nu';
```

```
    Description[EXPIVOTLib_TLB.exFilterBarOr] := 'sau';
    Import('C:\Program Files\Exontrol\ExPivot\Sample\data.txt',Null);
    DataColumns.Item['OrderDate'].SortType := EXPIVOTLib_TLB.SortDate;
    PivotRows := '9';
    Refresh();
end
```

VFP

```
with thisform.Pivot1
    DEBUGOUT( .Description(17) )
    .Object.Description(17) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August
Septembrie Octombrie Noiembrie Decembrie"
    .Object.Description(0) = "(Toate)"
    .Object.Description(1) = "(Gol)"
    .Object.Description(2) = "(Plin)"
    .Object.Description(3) = "Filtreaza:"
    .Object.Description(16) = "Azi"
    .Object.Description(18) = "D L Ma Mi J V S"
    .Object.Description(25) = "Exclud"
    .Object.Description(26) = "Coloane"
    .Object.Description(11) = "si"
    .Object.Description(12) = "Data:"
    .Object.Description(15) = "Data"
    .Object.Description(13) = "la"
    .Object.Description(24) = "nu"
    .Object.Description(23) = "sau"
    .Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    .DataColumns.Item("OrderDate").SortType = 2
    .PivotRows = "9"
    .Refresh
endwith
```

dBASE Plus

```
local oPivot

oPivot = form.EXPIVOTACTIVEXCONTROL1.nativeObject
```

```
? oPivot.Description(17)
oPivot.Template = [Description(17) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie"] // oPivot.Description(17) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie"
oPivot.Template = [Description(0) = "(Toate)"] // oPivot.Description(0) = "(Toate)"
oPivot.Template = [Description(1) = "(Gol)"] // oPivot.Description(1) = "(Gol)"
oPivot.Template = [Description(2) = "(Plin)"] // oPivot.Description(2) = "(Plin)"
oPivot.Template = [Description(3) = "Filtreaza:"] // oPivot.Description(3) = "Filtreaza:"
oPivot.Template = [Description(16) = "Azi"] // oPivot.Description(16) = "Azi"
oPivot.Template = [Description(18) = "D L Ma Mi J V S"] // oPivot.Description(18) = "D L Ma Mi J V S"
oPivot.Template = [Description(25) = "Exclud"] // oPivot.Description(25) = "Exclud"
oPivot.Template = [Description(26) = "Coloane"] // oPivot.Description(26) = "Coloane"
oPivot.Template = [Description(11) = "si"] // oPivot.Description(11) = "si"
oPivot.Template = [Description(12) = "Data:"] // oPivot.Description(12) = "Data:"
oPivot.Template = [Description(15) = "Data"] // oPivot.Description(15) = "Data"
oPivot.Template = [Description(13) = "la"] // oPivot.Description(13) = "la"
oPivot.Template = [Description(24) = "nu"] // oPivot.Description(24) = "nu"
oPivot.Template = [Description(23) = "sau"] // oPivot.Description(23) = "sau"
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.DataColumns.Item("OrderDate").SortType = 2
oPivot.PivotRows = "9"
oPivot.Refresh()
```

XBasic (Alpha Five)

```
Dim oPivot as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
? oPivot.Description(17)
oPivot.Template = "Description(17) = `Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie`" // oPivot.Description(17) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie"
```

```
oPivot.Template = "Description(0) = `(Toate)`" // oPivot.Description(0) = "(Toate)"
oPivot.Template = "Description(1) = `(Gol)`" // oPivot.Description(1) = "(Gol)"
oPivot.Template = "Description(2) = `(Plin)`" // oPivot.Description(2) = "(Plin)"
oPivot.Template = "Description(3) = `Filtreaza:`" // oPivot.Description(3) = "Filtreaza:"
oPivot.Template = "Description(16) = `Azi`" // oPivot.Description(16) = "Azi"
oPivot.Template = "Description(18) = `D L Ma Mi J V S`" // oPivot.Description(18) =
"D L Ma Mi J V S"
oPivot.Template = "Description(25) = `Exclud`" // oPivot.Description(25) = "Exclud"
oPivot.Template = "Description(26) = `Coloane`" // oPivot.Description(26) =
"Coloane"
oPivot.Template = "Description(11) = `si`" // oPivot.Description(11) = "si"
oPivot.Template = "Description(12) = `Data:`" // oPivot.Description(12) = "Data:"
oPivot.Template = "Description(15) = `Data`" // oPivot.Description(15) = "Data"
oPivot.Template = "Description(13) = `la`" // oPivot.Description(13) = "la"
oPivot.Template = "Description(24) = `nu`" // oPivot.Description(24) = "nu"
oPivot.Template = "Description(23) = `sau`" // oPivot.Description(23) = "sau"
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.DataColumns.Item("OrderDate").SortType = 2
oPivot.PivotRows = "9"
oPivot.Refresh()
```

Visual Objects

```
OutputDebugString(String2Psz( oDCOCX_Exontrol1:
[Description,exFilterBarDateMonths] ))
oDCOCX_Exontrol1:[Description,exFilterBarDateMonths] := "Ianuarie Februarie Martie
Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie"
oDCOCX_Exontrol1:[Description,exFilterBarAll] := "(Toate)"
oDCOCX_Exontrol1:[Description,exFilterBarBlanks] := "(Gol)"
oDCOCX_Exontrol1:[Description,exFilterBarNonBlanks] := "(Plin)"
oDCOCX_Exontrol1:[Description,exFilterBarFilterForCaption] := "Filtreaza:"
oDCOCX_Exontrol1:[Description,exFilterBarDateTodayCaption] := "Azi"
oDCOCX_Exontrol1:[Description,exFilterBarDateWeekDays] := "D L Ma Mi J V S"
oDCOCX_Exontrol1:[Description,exFilterBarExclude] := "Exclud"
oDCOCX_Exontrol1:[Description,exColumnsFloatBar] := "Coloane"
```

```
oDCOCX_Exontrol1:[Description,exFilterBarAnd] := "si"
oDCOCX_Exontrol1:[Description,exFilterBarDate] := "Data:"
oDCOCX_Exontrol1:[Description,exFilterBarDateTitle] := "Data"
oDCOCX_Exontrol1:[Description,exFilterBarDateTo] := "la"
oDCOCX_Exontrol1:[Description,exFilterBarNot] := "nu"
oDCOCX_Exontrol1:[Description,exFilterBarOr] := "sau"
oDCOCX_Exontrol1:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt",nil)
oDCOCX_Exontrol1:DataColumns:[Item,"OrderDate"]:SortType := SortDate
oDCOCX_Exontrol1:PivotRows := "9"
oDCOCX_Exontrol1:Refresh()
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
MessageBox("Information",string( oPivot.Description(17) ))
oPivot.Description(17,"Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August
Septembrie Octombrie Noiembrie Decembrie")
oPivot.Description(0,"(Toate)")
oPivot.Description(1,"(Gol)")
oPivot.Description(2,"(Plin)")
oPivot.Description(3,"Filtreaza:")
oPivot.Description(16,"Azi")
oPivot.Description(18,"D L Ma Mi J V S")
oPivot.Description(25,"Exclud")
oPivot.Description(26,"Coloane")
oPivot.Description(11,"si")
oPivot.Description(12,"Data:")
oPivot.Description(15,"Data")
oPivot.Description(13,"la")
oPivot.Description(24,"nu")
oPivot.Description(23,"sau")
oPivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
oPivot.DataColumns.Item("OrderDate").SortType = 2
oPivot.PivotRows = "9"
```

```
oPivot.Refresh()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   ShowIn (ComDescription(Self,OLEexFilterBarDateMonths))
   Set ComDescription OLEexFilterBarDateMonths to "Ianuarie Februarie Martie
Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie"
   Set ComDescription OLEexFilterBarAll to "(Toate)"
   Set ComDescription OLEexFilterBarBlanks to "(Gol)"
   Set ComDescription OLEexFilterBarNonBlanks to "(Plin)"
   Set ComDescription OLEexFilterBarFilterForCaption to "Filtreaza:"
   Set ComDescription OLEexFilterBarDateTodayCaption to "Azi"
   Set ComDescription OLEexFilterBarDateWeekDays to "D L Ma Mi J V S"
   Set ComDescription OLEexFilterBarExclude to "Exclud"
   Set ComDescription OLEexColumnsFloatBar to "Coloane"
   Set ComDescription OLEexFilterBarAnd to "si"
   Set ComDescription OLEexFilterBarDate to "Data:"
   Set ComDescription OLEexFilterBarDateTitle to "Data"
   Set ComDescription OLEexFilterBarDateTo to "la"
   Set ComDescription OLEexFilterBarNot to "nu"
   Set ComDescription OLEexFilterBarOr to "sau"
   Get ComImport "C:\Program Files\Exontrol\ExPivot\Sample\data.txt" Nothing to
Nothing
   Variant voColumns
   Get ComDataColumns to voColumns
   Handle hoColumns
   Get Create (RefClass(cComColumns)) to hoColumns
   Set pvComObject of hoColumns to voColumns
      Variant voColumn
      Get ComItem of hoColumns "OrderDate" to voColumn
      Handle hoColumn
      Get Create (RefClass(cComColumn)) to hoColumn
      Set pvComObject of hoColumn to voColumn
         Set ComSortType of hoColumn to OLESortDate
```

```
      Send Destroy to hoColumn
   Send Destroy to hoColumns
   Set ComPivotRows to "9"
   Send ComRefresh
End_Procedure
```

XBase++

```
#include "AppEvent.ch"
#include "ActiveX.ch"

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oPivot

   oForm := XbpDialog():new( AppDesktop() )
   oForm:drawingArea:clipChildren := .T.
   oForm:create( ,,{100,100}, {640,480},, .F. )
   oForm:close  := {|| PostAppEvent( xbeP_Quit )}

   oPivot := XbpActiveXControl():new( oForm:drawingArea )
   oPivot:CLSID  := "Exontrol.Pivot.1" /*{5C9DF3D3-81B1-42C4-BED6-
658F17748686}*/
   oPivot:create(,, {10,60},{610,370} )

      DevOut( oPivot:Description(17/*exFilterBarDateMonths*/) )
      oPivot:SetProperty("Description",17/*exFilterBarDateMonths*/,"Ianuarie
Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie
Decembrie")
      oPivot:SetProperty("Description",0/*exFilterBarAll*/,"(Toate)")
      oPivot:SetProperty("Description",1/*exFilterBarBlanks*/,"(Gol)")
      oPivot:SetProperty("Description",2/*exFilterBarNonBlanks*/,"(Plin)")
      oPivot:SetProperty("Description",3/*exFilterBarFilterForCaption*/,"Filtreaza:")
      oPivot:SetProperty("Description",16/*exFilterBarDateTodayCaption*/,"Azi")
      oPivot:SetProperty("Description",18/*exFilterBarDateWeekDays*/,"D L Ma Mi J V
S")
```

```
      oPivot:SetProperty("Description",25/*exFilterBarExclude*/,"Exclud")
      oPivot:SetProperty("Description",26/*exColumnsFloatBar*/,"Coloane")
      oPivot:SetProperty("Description",11/*exFilterBarAnd*/,"si")
      oPivot:SetProperty("Description",12/*exFilterBarDate*/,"Data:")
      oPivot:SetProperty("Description",15/*exFilterBarDateTitle*/,"Data")
      oPivot:SetProperty("Description",13/*exFilterBarDateTo*/,"la")
      oPivot:SetProperty("Description",24/*exFilterBarNot*/,"nu")
      oPivot:SetProperty("Description",23/*exFilterBarOr*/,"sau")
      oPivot:Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
      oPivot:DataColumns:Item("OrderDate"):SortType := 2/*SortDate*/
      oPivot:PivotRows := "9"
      oPivot:Refresh()

   oForm:Show()
   DO WHILE nEvent != xbeP_Quit
      nEvent := AppEvent( @mp1, @mp2, @oXbp )
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO
RETURN
```

property Pivot.DisplayFilterList as FilterListEnum

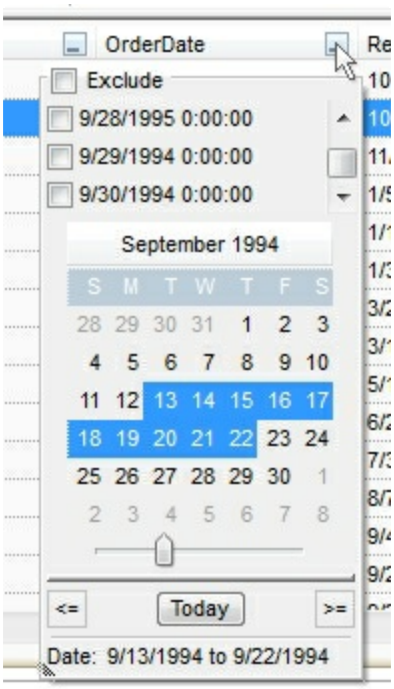
Specifies what the column's filter displays.

Type	Description
FilterListEnum	A FilterListEnum expression that specifies the options for filtering box being shown when the user clicks the drop down filter button.

By default, the DisplayFilterList property is exFilterListDefault, which is a bit-OR combination of : exAllItems | exSortItemsAsc | exShowCheckBox | exShowFocusItem | exShowExclude. For instance, you can use the DisplayFilterList property to allow displaying the drop down filter in the column's header, if the DisplayFilterList property is exNoItems(2), or you can hide the Exclude field, by setting the DisplayFilterList property on exAllItems | exSortItemsAsc | exShowCheckBox | exShowFocusItem. The FilterInclude property determines the items to be shown when the user applies a filter. The FilterCriteria property determines whether the OR or AND is used between columns. You can use the FilterBarPromptVisible property to show or hide the control's filter prompt field. The SortType property of the Column object determines the type of the filtering box being displayed as shown bellow:

The following screen shot shows the filtering box, if the SortType property is SortString:

The following screen shot shows the filtering box, if the SortType property is SortDate:


OrderDate
Exclude
9/28/1995 0:00:00
9/29/1994 0:00:00
9/30/1994 0:00:00
September 1994
S M T W T F S
28 29 30 31 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 1
2 3 4 5 6 7 8
<=
Today
>=
Date: 9/13/1994 to 9/22/1994

property Pivot.DisplayPivotData as Long

Retrieves or sets the maximum number of rows to be displayed from the pivot's data.

Type	Description
Long	A Long expression that specifies the number of rows to be displayed on the control's list once the data is loaded. If negative, the control displays all loaded rows.

By default, the DisplayPivotData property is 256, which means that the control displays max 256 rows in the control's list (for the data being imported). This property does not affect the number of items / rows to be loaded from the DATA you provide , it just adjusts the number of rows to be displayed on the control's list. If negative, all items/rows are being displayed on the control's list. You can use the property to prevent displaying the entire data once the data is loaded to the control (this could be a time consuming if you load a large data). The ... (three dots) is on the last line, indicating that there is more data that can be displayed The DisplayPivotFields property specifies the number of maximum columns to be added during the execution of the current layout. The LoadHeadersOnly property loads the headers only, so no data is loaded.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method

*The DisplayPivotData property **has effect only** if calling before any of the previously methods. This property does not affect the control's list once the control display grouping data (the pivot bar shows no empty content). In other words, the DisplayPivotData property affects only the data to be displayed once new data is drop to the control. Use the DisplayPivotRows property to limit the number of generated rows (PivotRows property is not empty).*

The following screen shot shows the control's list if the DisplayPivotData property is 6:

T...

ShipCountry	ShipRegion	ShipCity	ShipNa
Germany		Cunewalde	QUICK
USA	NM	Albuquerque	Rattles
USA	OR	Portland	Lones
Germany		Stuttgart	Die Wa
Mexico		México D.F.	Pericle
Switzerland		Bern	Chop-
...	...	...	

property Pivot.DisplayPivotFields as Long

Retrieves or sets the maximum number of columns to be displayed on the control's list.

Type	Description
Long	A Long expression that specifies the number of columns to be to be added during the execution of the current layout. If negative all executing columns are shown in the control's list

By default, the The DisplayPivotFields property is 256. The DisplayPivotFields property specifies the number of maximum columns to be added during the execution of the current layout. The DisplayPivotData property retrieves or sets the maximum number of rows to be displayed from the pivot's data. For instance, this property is useful to limit the columns to be added to the control's list, when you are grouping more columns, which can generate a lot of columns to be shown. This property prevent showing all of them during executing of the layout. The PivotColumns property indicates the list of data columns to be shown on the control's list. The DisplayPivotRows property retrieves or sets the maximum number of rows to be generated on the control's list.

property Pivot.DisplayPivotRows as Long

Retrieves or sets the maximum number of rows to be generated on the control's list.

Type	Description
Long	A Long expression that specifies the number of rows when the control generates the result.

By default, the DisplayPivotRows property is 16384, which indicates that the number of rows is limited to 16384 rows. If the DisplayPivotRows property is -1, there is no limit for generated rows. The DisplayPivotFields property specifies the number of maximum columns to be added during the execution of the current layout. The DisplayPivotData property retrieves or sets the maximum number of rows to be displayed from the pivot's data.

property Pivot.DrawGridLines as GridLinesEnum

Retrieves or sets a value that indicates whether the grid lines are visible or hidden.

Type	Description
GridLinesEnum	A GridLinesEnum expression that specifies the grid lines to be shown on the control's list.

Use the DrawGridLines property to add grid lines to the current view. Use the GridLineColor property to specify the color for grid lines. The GridLineStyle property to specify the style for horizontal or/and vertical gridlines in the control. Use the LinesAtRoot property specifies whether the control links the root items of the control. Use the HasLines property to specify whether the control draws the link between child items to their corresponding parent item

property Pivot.Enabled as Boolean

Enables or disables the control.

Type	Description
Boolean	A boolean expression that indicates whether the control is enabled or disabled.

Use the Enabled property to disable the control. Use the ForeColor property to change the control's foreground color. Use the BackColor property to change the control's background color.

method Pivot.EndUpdate ()

Resumes painting the control after painting is suspended by the BeginUpdate method.

Type	Description

The BeginUpdate and EndUpdate methods increases the speed of loading your events, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too. You can use the Refresh method to refresh the control's content.

property Pivot.EventParam(Parameter as Long) as Variant

Retrieves or sets a value that indicates the current's event parameter.

Type	Description
Parameter as Long	A long expression that indicates the index of the parameter being requested ie 0 means the first parameter, 1 means the second, and so on. If -1 is used the EventParam property retrieves the number of parameters. Accessing an not-existing parameter produces an OLE error, such as invalid pointer (E_POINTER)
Variant	A VARIANT expression that specifies the parameter's value.

The EventParam method is provided to allow changing the event's parameters passed by reference, even if your environment does not support changing it (uniPaas 1.5 (formerly known as eDeveloper), DBase, and so on). For instance, Unipaas event-handling logic cannot update ActiveX control variables by updating the received arguments. The EventParam(0) retrieves the value of the first parameter of the event, while the EventParam(1) = 0, changes the value of the second parameter to 0 (the operation is successfully, only if the parameter is passed by reference). The EventParam(-1) retrieves the number of the parameters of the current event.

Let's take the event "event KeyDown (**KeyCode** as Integer, ByVal Shift as Integer)", where the KeyCode parameter is passed by reference. For instance, put the KeyCode parameter on 0, and the arrow keys are disabled while the control has the focus.

In most languages you will type something like:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
   KeyCode = 0
End Sub
```

In case your environment does not support events with parameters by reference, you can use a code like follows:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
   Control1.EventParam(0) = 0
End Sub
```

In other words, the EventParam property provides the parameters of the current event for reading or writing access, even if your environment does not allow changing parameters by

reference.

Calling the EventParam property outside of an event produces an OLE error, such as pointer invalid, as its scope was designed to be used only during events.

method Pivot.ExecuteTemplate (Template as String)

Executes a template and returns the result.

Type	Description
Template as String	A Template string being executed

Return	Description
Variant	A Variant expression that indicates the result after executing the Template.

Use the ExecuteTemplate property to returns the result of executing a template file. Use the Template property to execute a template without returning any result. Use the ExecuteTemplate property to execute code by passing instructions as a string (template string). For instance, you can use the EXPRINT.PrintExt = CONTROL.ExecuteTemplate("me") to print the control's content.

For instance, the following sample retrieves the the handle of the first visible item:

```
Debug.Print Pivot1.ExecuteTemplate("Items.FirstVisibleItem()")
```

Most of our UI components provide a Template page that's accessible in design mode. No matter what programming language you are using, you can have a quick view of the component's features using the WYSWYG Template editor.

- Place the control to your form or dialog.
- Locate the Properties item, in the control's context menu, in design mode. If your environment doesn't provide a Properties item in the control's context menu, please try to locate in the Properties browser.
- Click it, and locate the Template page.
- Click the Help button. In the left side, you will see the component, in the right side, you will see a x-script code that calls methods and properties of the control.

The control's Template page helps user to initialize the control's look and feel in design mode, using the x-script language that's easy and powerful. The Template page displays the control on the left side of the page. On the right side of the Template page, a simple editor is displayed where user writes the initialization code. The control's look and feel is automatically updated as soon as the user types new instructions. The Template script is saved to the container persistence (when Apply button is pressed), and it is executed when the control is initialized at runtime. Any component that provides a WYSWYG Template page, provides a Template property. The Template property executes code from a string (template string).

The Template script is composed by lines of instructions. Instructions are separated by

"\n\r" (newline) characters.

An instruction can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)…. *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The Template supports the following general functions:

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

method Pivot.ExpandAll ()

Expands all rows.

Type	Description

property Pivot.ExpandOnDblClick as Boolean

Specifies whether the item is expanded or collapsed if the user dbl clicks the item.

Type	Description
Boolean	A boolean expression that indicates whether an item is expanded on dbl click.

By default, the ExpandOnDblClick property is TrueUse the ExpandOnDblClick property to disable expanding or collapsing items when user dbl clicks an item. The control fires the DblClick event when user double clicks the control. Use the ShowDataOnDblClick property on True, to let the user to display the data that generated the result when a cell is double clicked.

method Pivot.Export ([Destination as Variant], [Options as Variant])

Exports the control's data to a CSV or HTML format.

Type	Description
Destination as Variant	A String expression that specifies the file/format to be created. The Destination parameter indicates the format to be created as follows: • if "htm" or "html", the control returns the HTML format (including CSS style) • Any file-name that ends on ".htm" or ".html" creates the file with the HTML format inside • missing, empty, or any other case the Export exports the control's data in CSV format. No error occurs, if the Export method can not create the file.
Options as Variant	A String expression that specifies the options to be used when exporting the control's data, as explained bellow.

Return	Description
Variant	A String expression that indicates the format being exported. It could be CSV or HTML format based on the Destination parameter.

The Export method can export the control's DATA to a CSV or HTML format. The Export method can export a collection of columns from selected, visible, check or all items. By default, the control export all items, unless there is no filter applied on the control, where only visible items are exported. No visual appearance is saved in CSV format, instead the HTML format includes the visual appearance in CSS style.

The following file samples, shows the format the Export method can export the control's DATA:

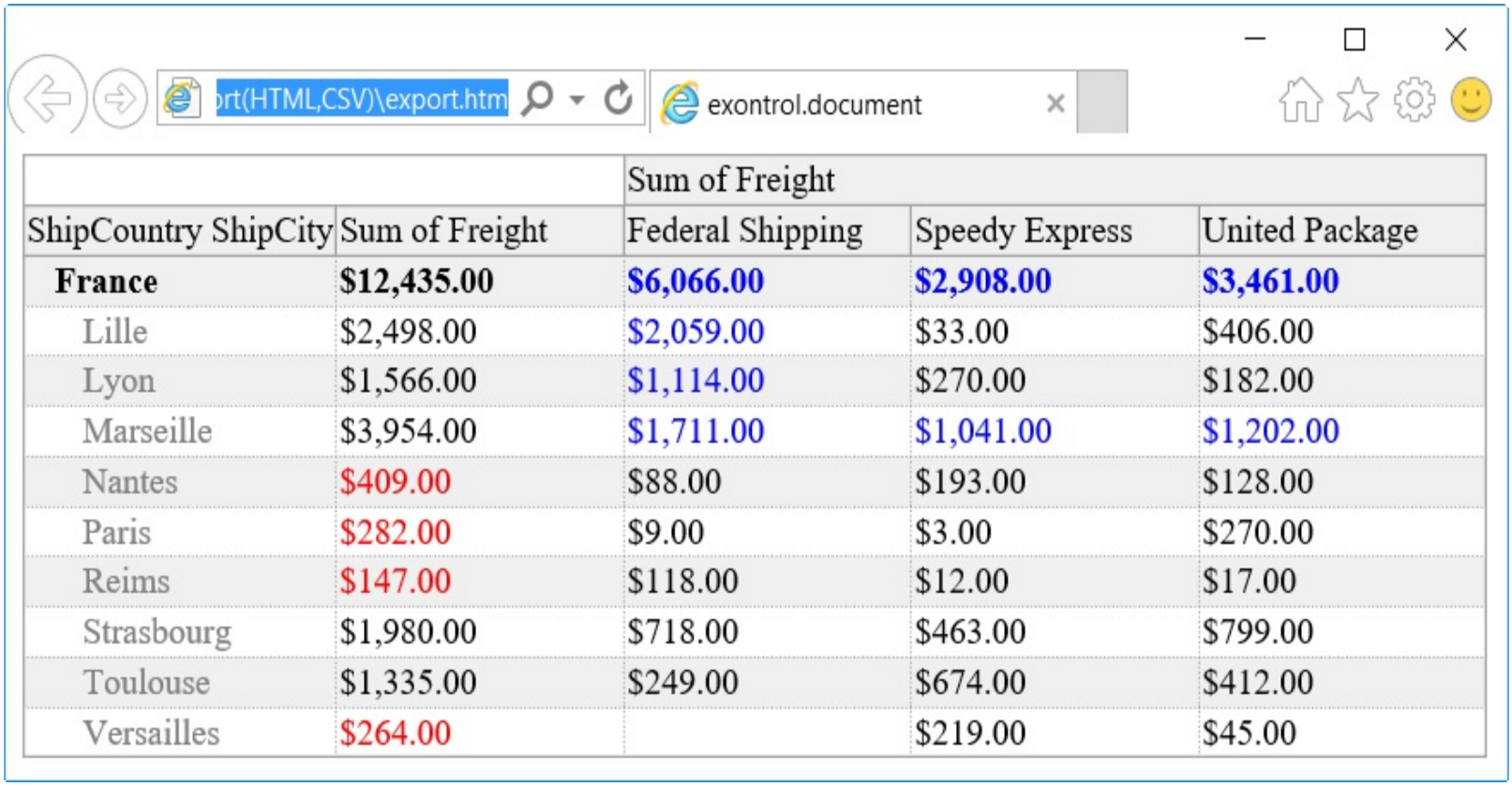
- CSV format
- HTML format

The following screen shot shows the control's DATA in CSV format:

export.txt - Microsoft Excel

	A	B	C	D	E
1			Sum of Freight		
2	ShipCount	Sum of Freight	Federal Shipping	Speedy Express	United Package
3	France	$12,435.00	$6,066.00	$2,908.00	$3,461.00
4	Lille	$2,498.00	$2,059.00	$33.00	$406.00
5	Lyon	$1,566.00	$1,114.00	$270.00	$182.00
6	Marseille	$3,954.00	$1,711.00	$1,041.00	$1,202.00
7	Nantes	$409.00	$88.00	$193.00	$128.00
8	Paris	$282.00	$9.00	$3.00	$270.00
9	Reims	$147.00	$118.00	$12.00	$17.00
10	Strasbourg	$1,980.00	$718.00	$463.00	$799.00
11	Toulouse	$1,335.00	$249.00	$674.00	$412.00
12	Versailles	$264.00		$219.00	$45.00

The following screen shot shows the control's DATA in HTML format:



ort(HTML,CSV)\export.htm

exontrol.document

		Sum of Freight		
ShipCountry ShipCity	Sum of Freight	Federal Shipping	Speedy Express	United Package
France	**$12,435.00**	**$6,066.00**	**$2,908.00**	**$3,461.00**
Lille	$2,498.00	$2,059.00	$33.00	$406.00
Lyon	$1,566.00	$1,114.00	$270.00	$182.00
Marseille	$3,954.00	$1,711.00	$1,041.00	$1,202.00
Nantes	$409.00	$88.00	$193.00	$128.00
Paris	$282.00	$9.00	$3.00	$270.00
Reims	$147.00	$118.00	$12.00	$17.00
Strasbourg	$1,980.00	$718.00	$463.00	$799.00
Toulouse	$1,335.00	$249.00	$674.00	$412.00
Versailles	$264.00		$219.00	$45.00

The Options parameter consists a list of fields separated by | character, in the following order:

1. The first field could be **all**, **vis**, **sel** or **chk**, to export all, just visible, selected or checked items. The all option is used, if the field is missing. The **all** option displays all items, including the hidden or collapsed items. The **vis** option includes the visible items only, not including the child items of a collapsed item, or not-visible items (item's height is 0). The **sel** options lists the items being selected. The **chk** option lists all check and visible items. If chk option is used, the first column in the columns list should indicate the index of the column being queried for a check box state.
2. the second field indicates the column to be exported. All visible columns are exported, if missing. The list of columns is separated by , character, and indicates the index of the column to be shown on the exported data. The first column in the list indicates the column being queried, if the option **chk** is used.
3. the third field indicates the character to separate the fields inside each exported line [tab character-if missing]. This field is valid, only when exporting to a CSV format
4. the forth field could be **ansi** or **unicode**, which indicates the character-set to save the control's content to Destination. For instance, Export(Destination,"|||unicode") saves the control's content to destination in UNICODE format (two-bytes per character). By default, the Export method creates an ANSI file (one-byte character)

The Destination parameter indicates the file to be created where exported date should be saved. For instance, Export("c:\temp\export.html") exports the control's DATA to export.html file in HTML format, or Export("","sel|0,1|;") returns the cells from columns 0, 1 from the selected items, to a CSV format using the ; character as a field separator.

The "CSV" refers to any file that:

- CSV stands for Comma Separated Value
- is plain text using a character set such as ASCII, Unicode,
- consists of records (typically one record per line),
- with the records divided into fields separated by delimiters (typically a single reserved character such as tab, comma, or semicolon; sometimes the delimiter may include optional spaces),
- where every record has the same sequence of fields

The "HTML" refers to any file that:

- HTML stands for HyperText Markup Language.
- is plain text using a character set such as ASCII, Unicode
- It's the way web pages are encoded to handle things like bold, italics and even color text red.

You can use the Copy/CopyTo to export the control's view to clipboard/EMF/BMP/JPG/PNG/GIF or PDF format.

Currently, the /COM version allows you to use the Export method in window-less mode (no

user interface). In order to use the control in window-less mode, after creation of the /COM object it is required to call any of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method.

Any of the following templates creates dynamically the /COM component, and uses the Export method to display the result of imported data:

```
CreateObject("Exontrol.Pivot")
{
   Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
   PivotRows = "0"
   PivotColumns = "avg(5)"
   print(Export())
}
```

or:

```
CreateObject("Exontrol.Pivot")
{
   Dim rs
   rs = CreateObject("ADOR.Recordset")
   {
      ' Change the Path to the SAMPLE.MDB if nothing is displayed
      Open("Data","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\Access2007\sample.accdb", 3, 3 )
   }
   DataSource = rs
   PivotRows = "0"
   PivotColumns = "max(5)"
   print(Export())
}
```

or:

```
CreateObject("Exontrol.Pivot")
{
   LoadXML("https://www.exontrol.net/testing.xml")
```

```
    PivotRows = "1"
    PivotColumns = "max(6)"
    print(Export())
}
```

You can use the eXHelper tool to convert any of these templates to your programming languages. For instance, in VB6 the code shows as:

```
With CreateObject("Exontrol.Pivot")
    .Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"
    .PivotRows = "0"
    .PivotColumns = "avg(5)"
    Debug.Print( .Export() )
End With
```

or in PowerBuilder shows as:

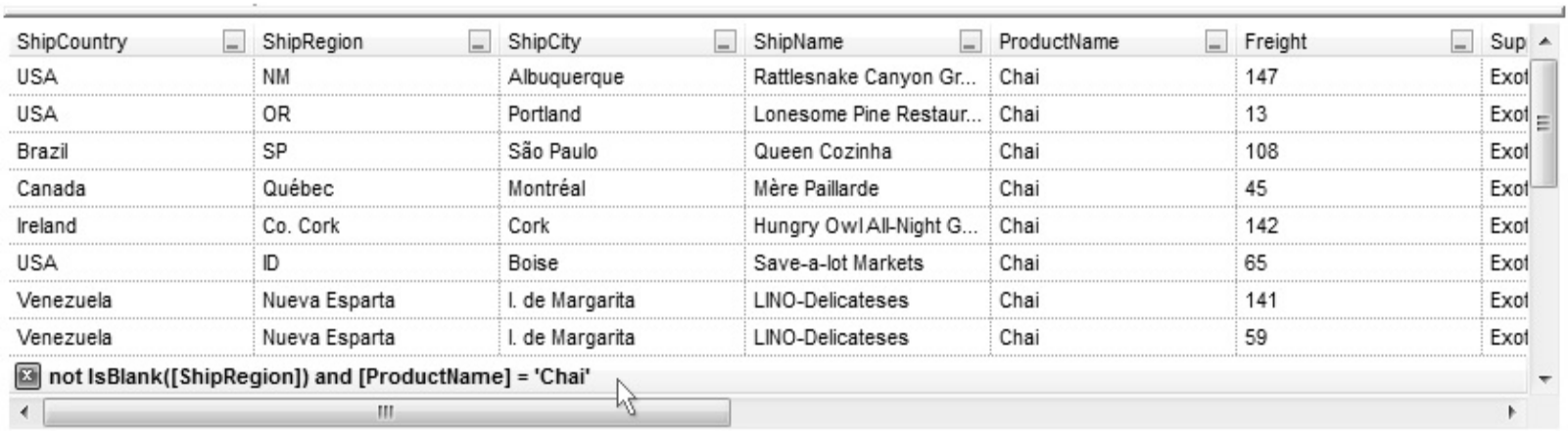
```
OleObject var_Pivot

var_Pivot = CREATE OLEObject
var_Pivot.ConnectToNewObject("Exontrol.Pivot")
    var_Pivot.Import("C:\Program Files\Exontrol\ExPivot\Sample\data.txt")
    var_Pivot.PivotRows = "0"
    var_Pivot.PivotColumns = "avg(5)"
    MessageBox("Information",string( String(var_Pivot.Export()) ))
```

property Pivot.FilterBarBackColor as Color

Specifies the background color of the control's filter bar.

Type	Description
Color	A color expression that defines the background color for description of the control's filter. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

Use the FilterBarForeColor and FilterBarBackColor properties to define the colors used to paint the description for control's filter. Use the FilterBarHeight property to hide the control's filter bar header. Use the BackColor property to specify the control's background color.



The following VB sample changes the visual appearance for control's filter bar. The sample applies the skin "" to the "close" button in the control's filter bar, and the "" skin to the filter bar caption area:

```
With Pivot1
  With .VisualAppearance
    .Add &H2, App.Path + "\fbarclose.ebn"
    .Add &H12, App.Path + "\filterbar.ebn"
  End With
  .Background(exFooterFilterBarButton) = &H2000000
  .FilterBarBackColor = &H12000000
  .FilterBarForeColor = RGB(255, 255, 255)
End With
```

ShipCountry	ShipRegion	ShipCity	ShipName	ProductName	Freight	Sup
USA	NM	Albuquerque	Rattlesnake Canyon Gr...	Chai	147	Exot
USA	OR	Portland	Lonesome Pine Restaur...	Chai	13	Exot
Brazil	SP	São Paulo	Queen Cozinha	Chai	108	Exot
Canada	Québec	Montréal	Mère Paillarde	Chai	45	Exot
Ireland	Co. Cork	Cork	Hungry Owl All-Night G...	Chai	142	Exot
USA	ID	Boise	Save-a-lot Markets	Chai	65	Exot
Venezuela	Nueva Esparta	I. de Margarita	LINO-Delicateses	Chai	141	Exot
Venezuela	Nueva Esparta	I. de Margarita	LINO-Delicateses	Chai	59	Exot

not IsBlank([ShipRegion]) and [ProductName] = 'Chai'

The following C++ sample changes the visual appearance for the "close" button in the control's filter bar:

```
#include "Appearance.h"
m_pivot.GetVisualAppearance().Add( 0x2,
COleVariant(_T("D:\\Temp\\ExPivot.Help\\fbarclose.ebn")) );
m_pivot.GetVisualAppearance().Add( 0x12,
COleVariant(_T("D:\\Temp\\ExPivot.Help\\filterbar.ebn")) );
m_pivot.SetBackground( 1 /*exFooterFilterBarButton*/, 0x2000000 );
m_pivot.SetFilterBarBackColor( 0x12000000 );
m_pivot.SetFilterBarForeColor( RGB(255,255,255) );
```

The following VB.NET sample changes the visual appearance for the "close" button in the control's filter bar:

```
With AxPivot1
   With .VisualAppearance
      .Add(&H2, "D:\Temp\ExPivot.Help\fbarclose.ebn")
      .Add(&H12, "D:\Temp\ExPivot.Help\filterbar.ebn")
   End With
   .Template = "FilterBarBackColor = 301989888"
   .FilterBarForeColor = Color.White
   .set_Background(EXTREELib.BackgroundPartEnum.exFooterFilterBarButton, &H2000000)
End With
```

The following C# sample changes the visual appearance for the "close" button in the control's filter bar:

```
axPivot1.VisualAppearance.Add(0x2, "D:\\Temp\\ExPivot.Help\\fbarclose.ebn");
axPivot1.VisualAppearance.Add(0x12, "D:\\Temp\\ExPivot.Help\\filterbar.ebn");
axPivot1.set_Background(EXTREELib.BackgroundPartEnum.exFooterFilterBarButton,
```

```
0x2000000);
axPivot1.Template = "FilterBarBackColor = 301989888";
axPivot1.FilterBarForeColor = Color.White;
```

The following VFP sample changes the visual appearance for the "close" button in the control's filter bar:

```
With thisform.Pivot1
   With .VisualAppearance
      .Add(2, "D:\Temp\ExPivot.Help\fbarclose.ebn")
      .Add(18, "D:\Temp\ExPivot.Help\filterbar.ebn")
   EndWith
   .Object.Background(1) = 33554432
   .FilterBarBackColor = 301989888
   .FilterBarForeColor = RGB(255,255,255)
EndWith
```

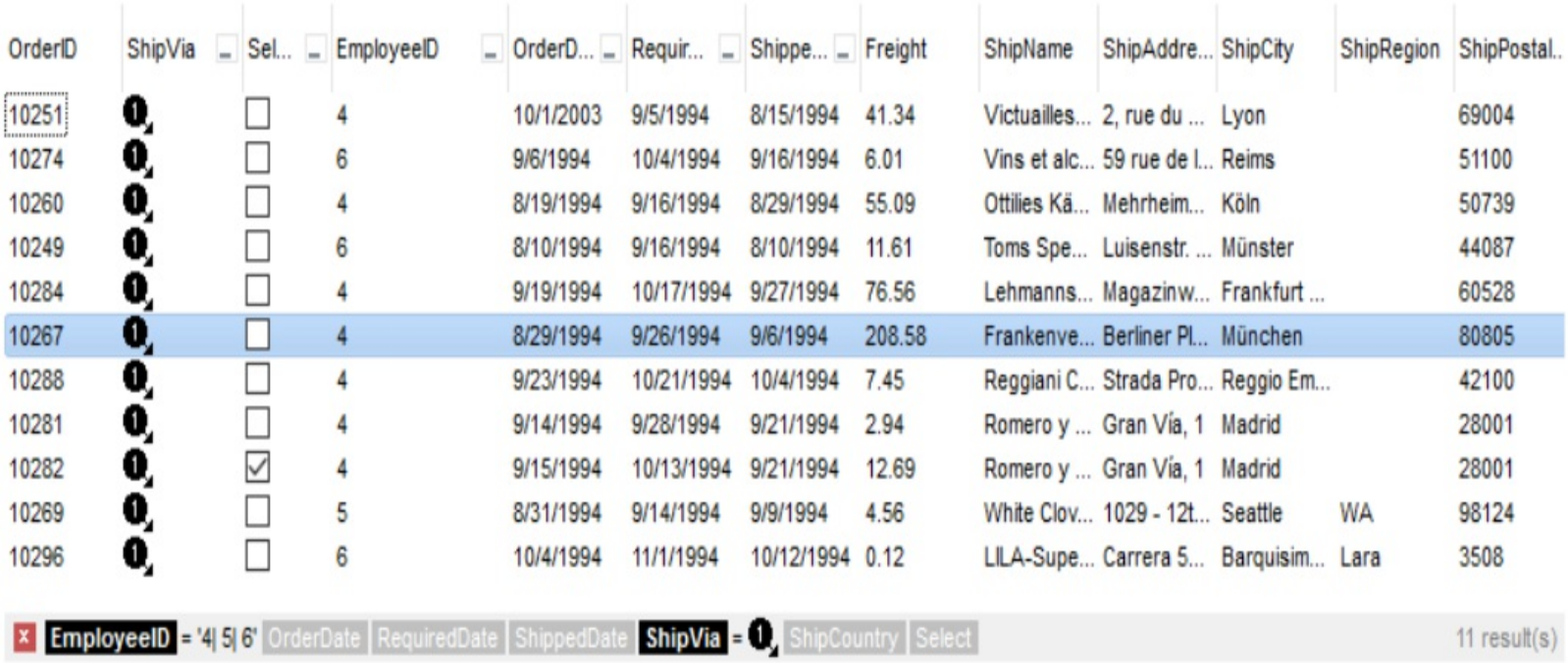
The 301989888 value is the 0x12000000 value in hexadecimal.

property Pivot.FilterBarCaption as String

Specifies the filter bar's caption.

Type	Description
String	A string value that defines the expression to display the control's filter bar.

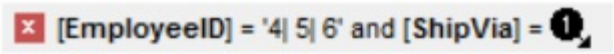
By default, the FilterBarCaption property is empty. You can use the FilterBarCaption property to define the way the filter bar's caption is being displayed. The FilterBarCaption is displayed on the bottom side of the control where the control's filter bar is shown. While the FilterBarCaption property is empty, the control automatically builds the caption to be displayed on the filter bar from all columns that participates in the filter using its name and values. For instance, if the control filters items based on the columns "EmployeeID" and "ShipVia", the control's filter bar caption would appear such as "[EmployeeID] = '...' and [ShipVia] = '...'". The FilterBarCaption property supports expressions as explained bellow.



For instance:

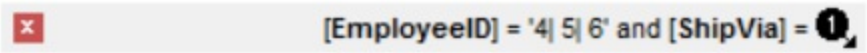
- "no filter", shows no filter caption all the time

- "" displays no filter bar, if no filter is applied, else it displays the current filter



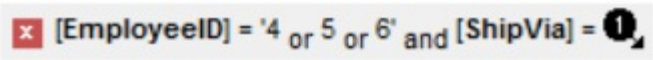
- "`<r>` + value", displays the current filter caption aligned to the right. You can include

the exFilterBarShowCloseOnRight flag into the FilterBarPromptVisible property to display the close button aligned to the right

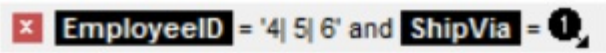


- "value replace ` and ` with `<fgcolor=FF0000> and </fgcolor>`", replace the AND keyword with a different foreground color

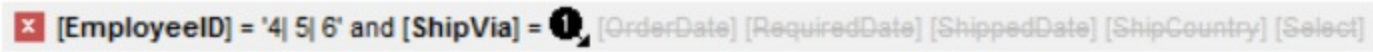
- "value replace ` and ` with `<off 4> and </off>` replace `|` with ` <off 4>or</off> ` replace ` ` with ` `", replaces the AND and | values



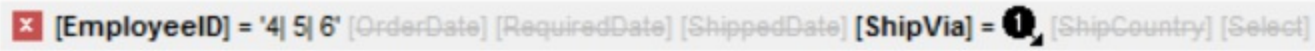
- "value replace `[` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `]` with ` </b></bgcolor></fgcolor>`", highlights the columns being filtered with a different background/foreground colors.



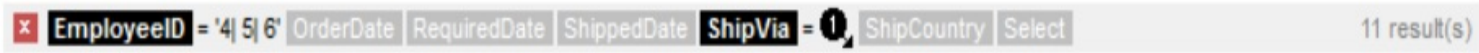
- "value + ` ` + available", displays the current filter, including all available columns to be filtered



- "allui" displays all available columns



- "((allui + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor></fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0> <fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all available columns to be filtered with different background/foreground colors including the number of items/results



Use the FilterBarForeColor and FilterBarBackColor properties to define the colors used to paint the description for control's filter. Use the FilterBarHeight property to specify the

height of the control's filter bar. Use the FilterBarFont property to specify the font for the control's filter bar. Use the Description property to define predefined strings in the filter bar caption. The FilterBarPromptVisible property specifies whether how/where the control's filter/prompt is shown.

The FilterBarCaption method supports the following keywords, constants, operators and functions:

- **value** or **current** keyword returns the current filter as a string. At runtime the value may return a string such as "[EmployeeID] = '4| 5| 6' and [ShipVia] = <img>1</img>", so the control automatically applies HTML format, which you can change it. For instance, "upper(value)" displays the caption in uppercase or "value replace `<b>` with `<fgcolor=808080>` replace `</b>` with `</fgcolor>`" displays the column's name with a different foreground color.
- **itemcount** keyword returns the total number of items. At runtime the itemcount is a positive integer that indicates the count of all items. For instance, "value + `<r> <fgcolor=808080>Total: ` + itemcount" includes in the filter bar the number of items aligned to the right.
- **visibleitemcount** keyword returns the number of visible items. At runtime, the visibleitemcount is a positive integer if no filter is applied, and negative if a filter is applied. If positive, it indicates the number of visible items. The visible items does not include child items of a collapsed item. If negative, a filter is applied, and the absolute value minus one, indicates the number of visible items after filter is applied. 0 indicates no visible items, while -1 indicates that a filter is applied, but no item matches the filter criteria. For instance, "value + `<r><fgcolor=808080>` + (visibleitemcount < 0 ? (`Result: ` + (abs(visibleitemcount) - 1)) : (`Visible: ` + visibleitemcount))" includes "Visible: " plus number of visible items, if no filter is applied or "Result: " plus number of visible items, if filter is applied, aligned to the right
- **matchitemcount** keyword returns the number of items that match the filter. At runtime, the matchitemcount is a positive integer if no filter is applied, and negative if a filter is applied. If positive, it indicates the number of items within the control. If negative, a filter is applied, and the absolute value minus one, indicates the number of matching items after filter is applied. A matching item includes its parent items, if the control's FilterInclude property allows including child items. 0 indicates no visible items, while -1 indicates that a filter is applied, but no item matches the filter criteria. For instance, "value + `<r><fgcolor=808080>` + (matchitemcount < 0 ? (`Result: ` + (abs(matchitemcount) - 1)) : (`Visible: ` + matchitemcount))" includes "Visible: " plus number of visible items, if no filter is applied or "Result: " plus number of macthing items, if filter is applied, aligned to the right
- **leafitemcount** keyword returns the number of leaf items. A leaf item is an item with no child items. At runtime, the leafitemcount is a positive number that computes the number of leaf items (expanded or collapsed). For instance, the "value + `<r> <fgcolor=808080><font ;6>` + leafitemcount" displays the number of leaf items aligned

to the right with a different font and foreground color.

- **promptpattern** returns the pattern in the filter bar's prompt, as a string. The FilterBarPromptPattern specifies the pattern for the filter prompt. The control's filter bar prompt is visible, if the exFilterBarPromptVisible flag is included in the FilterBarPromptVisible property.
- **available** keyword returns the list of columns that are not currently part of the control's filter, but are available to be filtered. At runtime, the available keyword may return a string such as "<fgcolor=C0C0C0>[<s>OrderDate</s>]<fgcolor> </fgcolor>[<s>RequiredDate</s>]<fgcolor> </fgcolor>[<s>ShippedDate</s>]<fgcolor> </fgcolor>[<s>ShipCountry</s>]<fgcolor> </fgcolor>[<s>Select</s>]</fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "value + ` ` + available", displays the current filter, including all available columns to be filtered. For instance, the "value + `<r>` + available replace `C0C0C0` with `FF0000`" displays the available columns aligned to the right with a different foreground color.
- **allui** keyword returns the list of columns that are part of the current filter and available columns to be filtered. At runtime, the allui keyword may return a string such as "[EmployeeID] = '4| 5| 6'<fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>OrderDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>RequiredDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>ShippedDate</s>]</fgcolor><fgcolor> </fgcolor>[ShipVia] = <img>1</img><fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>ShipCountry</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>Select</s>]</fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "allui", displays the current filter, including all available columns to be filtered. For instance, the "((allui + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor></fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0><fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all available columns to be filtered with different background/foreground colors including the number of items/results
- **all** keyword returns the list of all columns (visible or hidden). At runtime, the all keyword may return a string such as "<fgcolor=C0C0C0>[<s>OrderID</s>]</fgcolor><fgcolor> </fgcolor>[EmployeeID] = '4| 5| 6'<fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>OrderDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>RequiredDate</s>]</fgcolor><fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "all", displays the current filter, including all other columns. For instance, the "((all + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor></fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0>
<fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all

columns with different background/foreground colors including the number of items/results

Also, the FilterBarCaption property supports predefined constants and operators/functions as described here.

Also, the FilterBarCaption property supports HTML format as described here:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the

index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

  shadow

  or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

  outline anti-aliasing

property Pivot.FilterBarFont as IFontDisp

Retrieves or sets the font for control's filter bar.

Type	Description
IFontDisp	A font object that indicates the font used to paint the description for control's filter

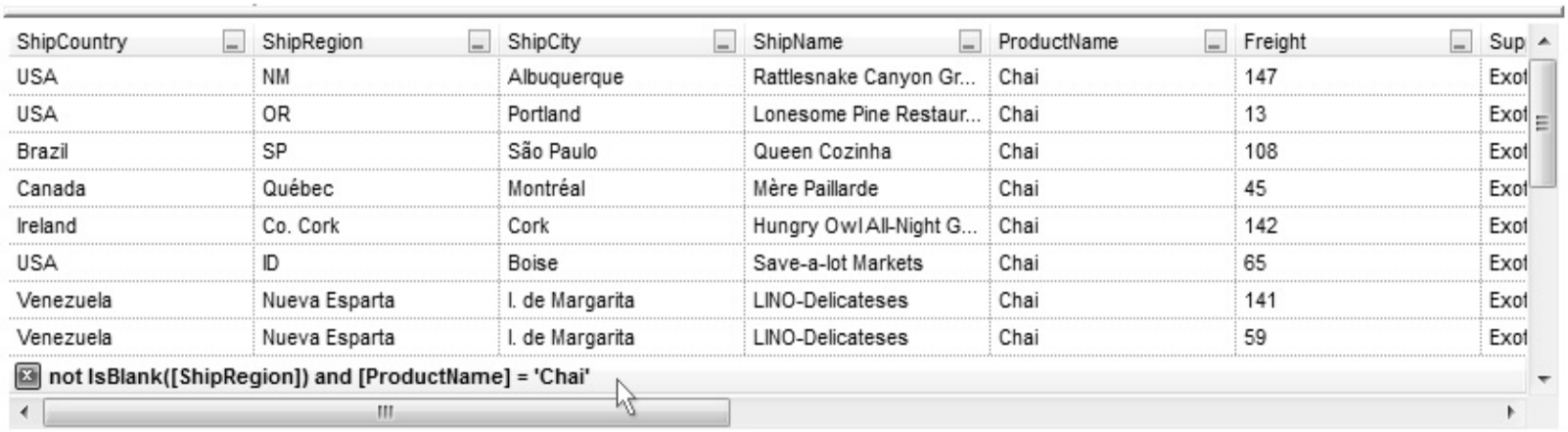
Use the FilterBarFont property to specify the font for the control's filter bar object. Use the Font property to set the control's font. Use the FilterBarHeight property to specify the height of the filter bar. Use the FilterBarCaption property to define the control's filter bar caption. Use the Refresh method to refresh the control.

property Pivot.FilterBarForeColor as Color

Specifies the foreground color of the control's filter bar.

Type	Description
Color	A color expression that defines the foreground color of the description of the control's filter.

Use the FilterBarForeColor and FilterBarBackColor properties to define colors used to paint the description of the control's filter. Use the FilterBarFont property to specify the filter bar's font. Use the FilterBarCaption property to specify the caption of the control's filter bar.



property Pivot.FilterBarHeight as Long

Specifies the height of the control's filter bar description.

Type	Description
Long	A long expression that indicates the height of the filter bar status.

The filter bar status defines the control's filter description. If the FilterBarHeight property is less than 0 the control automatically updates the height of the filter's description to fit in the control's client area. If the FilterBarHeight property is zero the filter's description is hidden. If the FilterBarHeight property is grater than zero it defines the height in pixels of the filter's description. Use the ClearFilter method to clear the control's filter. Use the FilterBarCaption property to define the control's filter bar caption. Use the FilterBarFont property to specify the font for the control's filter bar.

property Pivot.FilterBarPrompt as String

Specifies the caption to be displayed when the filter pattern is missing.

Type	Description
String	A string expression that indicates the HTML caption being displayed in the filter bar, when filter prompt pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature.

By default, the FilterBarPrompt property is "<fgcolor=808080>Start Filter...</fgcolor> ". The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature. Changing the FilterBarPrompt property won't change the current filter. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. Use the FilterBarCaption property to change the caption in the filter bar once a new filter is applied. The FilterBarFont property specifies the font to be used in the filter bar. The FilterBarBackColor property specifies the background color or the visual aspect of the control's filter bar. The FilterBarForeColor property specifies the foreground color or the control's filter bar.

The FilterBarPrompt property supports HTML format as described here:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The

HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

The FilterBarPrompt property has effect only if:

- FilterBarPromptVisible property is True
- FilterBarPromptPattern property is Empty.

property Pivot.FilterBarPromptColumns as Variant

Specifies the list of columns to be used when filtering using the prompt.

Type	Description
Variant	A long expression that indicates the index of the column to apply the filter prompt, a string expression that specifies the list of columns (indexes) separated by comma to apply the filter prompt, or a safe array of long expression that specifies the indexes of the columns to apply the filter. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area.

By default, the FilterBarPromptColumns property is -1. If the FilterBarPromptColumns property is -1, the filter prompt is applied for all columns, visible or hidden. Use the FilterBarPromptColumns property to specify the list of columns to apply the filter prompt pattern. The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar.

property Pivot.FilterBarPromptPattern as String

Specifies the pattern for the filter prompt.

Type	Description
String	A string expression that specifies the pattern to filter the list.

By default, the FilterBarPromptPattern property is empty. If the FilterBarPromptPattern property is empty, the filter bar displays the FilterBarPrompt property, if the FilterBarPromptVisible property is True. The FilterBarPromptPattern property indicates the patter to filter the list. The pattern may include wild characters if the FilterBarPromptType property is exFilterPromptPattern. The FilterBarPromptColumns specifies the list of columns to be used when filtering.

property Pivot.FilterBarPromptType as FilterPromptEnum

Specifies the type of the filter prompt.

Type	Description
FilterPromptEnum	A FilterPromptEnum expression that specifies how the items are being filtered.

By default, the FilterBarPromptType property is exFilterPromptContainsAll. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too. Use the FilterBarPromptVisible property to show the filter prompt. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. Use the FilterBarCaption property to change the caption in the filter bar once a new filter is applied.

The FilterBarPromptType property supports the following values:

- **exFilterPromptContainsAll**, The list includes the items that contains all specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptContainsAny**, The list includes the items that contains any of specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptStartWith**, The list includes the items that starts with any specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptEndWith**, The list includes the items that ends with any specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptPattern**, The filter indicates a pattern that may include wild characters to be used to filter the items in the list. The FilterBarPromptPattern property may include wild characters as follows:
  - '?' for any single character
  - '*' for zero or more occurrences of any character
  - '#' for any digit character
  - ' ' space delimits the patterns inside the filter

property Pivot.FilterBarPromptVisible as FilterBarVisibleEnum

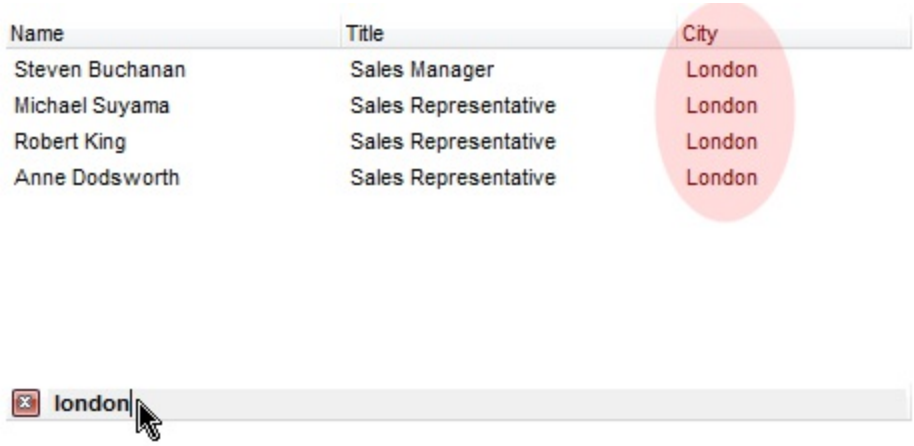
Shows or hides the control's filter bar including filter prompt.

Type	Description
FilterBarVisibleEnum	A FilterBarVisibleEnum expression that defines the way the control's filter bar is shown.

By default, The FilterBarPromptVisible property is exFilterBarHidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too. Use the FilterBarPromptVisible property to show the filter prompt. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. The FilterBarCaption property defines the caption to be displayed on the control's filter bar. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt.

The following screen show shows the filter prompt:

The following screen show shows the list once the user types "london":



property Pivot.FilterCriteria as String

Retrieves or sets the filter criteria.

Type	Description
String	A string expression that indicates the filter criteria.

By default, the FilterCriteria property is empty. Use the FilterCriteria property to specify whether you need to filter items using OR, NOT operators between columns. If the FilterCriteria property is empty, or not valid, the filter uses the AND operator between columns. Use the FilterCriteria property to specify how the items are filtered. Use the DisplayFilterList property to specify whether the column's header displays a drop down filter button. You can use the FilterBarPromptVisible property to show or hide the control's filter prompt field.

The FilterCriteria property supports the following operators:

- **not** operator (unary operator)
- **and** operator (binary operator)
- **or** operator (binary operator)

Use the **(** and **)** parenthesis to define the order execution in the clause, if case. The operators are grided in their priority order. The **%** character precedes the index of the column (zero based), and indicates the column. For instance, **%0 or %1** means that OR operator is used when both columns are used, and that means that you can filter for values that are in a column or for values that are in the second columns. If a column is not grided in the FilterCriteria property, and the user filters values by that column, the AND operator is used by default. For instance, let's say that we have three columns, and FilterCriteria property is "%0 or %1". If the user filter for all columns, the filter clause is equivalent with (%0 or %1) and %2, and it means all that match the third column, and is in the first or the second column.

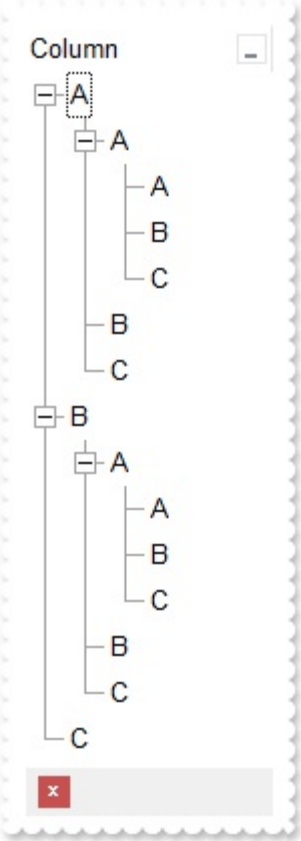
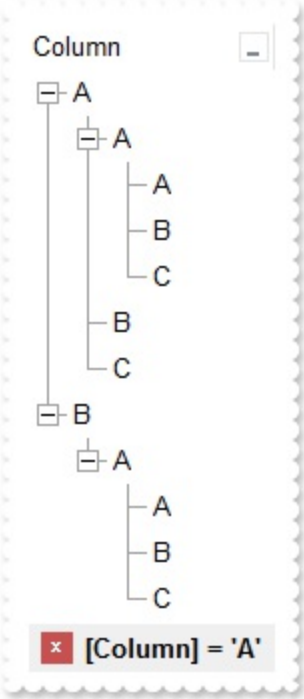
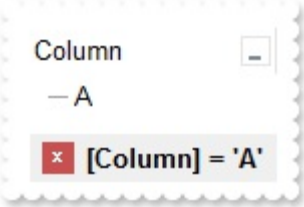
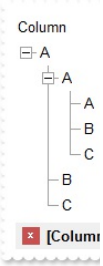
property Pivot.FilterInclude as FilterIncludeEnum

Specifies the items being included after the user applies the filter.

Type	Description
FilterIncludeEnum	A FilterIncludeEnum expression that indicates the items being included when the filter is applied.

By default, the FilterInclude property is exItemsWithoutChilds, which specifies that only items (and parent-items) that match the filter are being displayed. Use the FilterInclude property to specify whether the child- items should be displayed when the user applies the filter. Use the ClearFilter method to clear the control's filter. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the FilterBarPromptVisible property to show the control's filter-prompt, that allows you to filter items as you type.

The following table shows items to display, when filter for "A" items, using different values for FilterInclude property:

no filter	exItemsWithoutChilds 0	exItemsWithChilds 1	exRootsWithoutChilds 2	exRootsW 3
				

property Pivot.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object used to paint the items.

Use the Font property to change the control's font . Use the FilterBarFont property to assign a different font for the control's filter bar. Use the Refresh method to refresh the control. Use the BeginUpdate and EndUpdate method to maintain performance while perform multiple changes on the control.

The following VB sample assigns by code a new font to the control:

```
With Pivot1
   With .Font
      .Name = "Tahoma"
   End With
   .Refresh
End With
```

The following C++ sample assigns by code a new font to the control:

```
COleFont font = m_pivot.GetFont();
font.SetName( "Tahoma" );
m_pivot.Refresh();
```

the C++ sample requires definition of COleFont class (#include "Font.h")

The following VB.NET sample assigns by code a new font to the control:

```
With AxPivot1
   Dim font As System.Drawing.Font = New System.Drawing.Font("Tahoma", 10,
FontStyle.Regular, GraphicsUnit.Point)
   .Font = font
   .CtlRefresh()
End With
```

The following C# sample assigns by code a new font to the control:

```
System.Drawing.Font font = new System.Drawing.Font("Tahoma", 10, FontStyle.Regular);
```

```
axPivot1.Font = font;
axPivot1.CtlRefresh();
```

The following VFP sample assigns by code a new font to the control:

```
with thisform.Pivot1.Object
   .Font.Name = "Tahoma"
   .Refresh()
endwith
```

The following Template sample assigns by code a new font to the control:

```
Font
{
   Name = "Tahoma"
}
```

property Pivot.ForeColor as Color

Specifies the control's foreground color.

Type	Description
Color	A Color expression that specifies the control's foreground color.

The ForeColor property specifies the control's foreground color. The BackColor property specifies the control's background color. The ForeColor property changes the foreground color of the control's pivot bar. You can use the Background(exPivotBarForeColor) property to specify a different foreground color for the control's pivot bar. Use the Appearance property to specify the visual appearance of the control's frame.

property Pivot.ForeColorHeader as Color

Specifies the header's foreground color.

Type	Description
Color	A color expression that indicates the foreground color for control's header.

Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the Font property to change the control's font. Use the HeaderVisible property to hide the control's header bar.

method Pivot.FormatABC (Expression as String, [A as Variant], [B as Variant], [C as Variant])

Formats the A,B,C values based on the giving expression and returns the result.

Type	Description
Expression as String	A String that defines the expression to be evaluated.
A as Variant	A VARIANT expression that indicates the value of the A keyword.
B as Variant	A VARIANT expression that indicates the value of the B keyword.
C as Variant	A VARIANT expression that indicates the value of the C keyword.
Return	**Description**
Variant	A VARIANT expression that indicates the result of the evaluation the Pivot.

The FormatABC method formats the A,B,C values based on the giving expression and returns the result.

For instance:

- "A + B + C", adds / concatenates the values of the A, B and C
- "value MIN 0 MAX 99", limits the value between 0 and 99
- "value format ``", formats the value with two decimals, according to the control's panel setting
- "date(`now`)" returns the current time as double

The FormatABC method supports the following keywords, constants, operators and functions:

- **A** or **value** keyword, indicates a variable A whose value is giving by the A parameter
- **B** keyword, indicates a variable B whose value is giving by the B parameter
- **C** keyword, indicates a variable C whose value is giving by the C parameter

This property/method supports predefined constants and operators/functions as described here.

property Pivot.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

Type	Description
New as Boolean	Boolean expression that indicates whether to specify the anchors never clicked or anchors being clicked.
String	A String expression that indicates the HTMLformat to apply to anchor elements.

By default, the FormatAnchor(**True**) property is "<u><fgcolor=0000FF>#" that indicates that the anchor elements (that were never clicked) are underlined and shown in light blue. Also, the FormatAnchor(**False**) property is "<u><fgcolor=000080>#" that indicates that the anchor elements are underlined and shown in dark blue. *You can use the <a> anchor elements to insert hyperlinks to cells, bars or links.*

The visual effect is applied to the anchor elements, if the FormatAnchor property is not empty. For instance, if you want to do not show with a new effect the clicked anchor elements, you can use the FormatAnchor(**False**) = "", that means that the clicked or not-clicked anchors are shown with the same effect that's specified by FormatAnchor(**True**). An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the AnchorClick event to notify that the user clicks an anchor element. This event is fired only if prior clicking the control it shows the hand cursor. The AnchorClick event carries the identifier of the anchor, as well as application options that you can specify in the anchor element. The hand cursor is shown when the user hovers the mouse on the anchor elements.

property Pivot.FormatAppearances as FormatAppearances

Retrieves the FormatAppearances collection of the pivot control.

Type	Description
FormatAppearances	A FormatAppearances object that holds a collection of FormatAppearance objects to be displayed on the control's context menu.

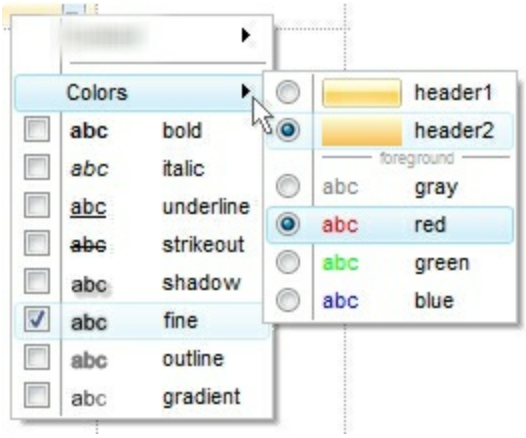
The FormatAppearances property gives access to a collection of FormatAppearance objects to be displayed on the control's context menu as seen bellow. Each FormatAppearance object contains font or color attributes that can be applied to any column/row on the control's list. If the PivotBarVisible property includes the exPivotBarAllowFormatAppearance flag, the control's context menu includes the FormatAppearance objects. If the exPivotBarAllowFormatAppearance flag is missing from the PivotBarVisible property, the control's context menu displays no FormatAppearance objects. The Add method adds a new FormatAppearance object to the collection. The FormatConditionalAppearances helps you to provide conditional-format for your data, or in other words, ability to highlight values that matches a specified expression.

The FormatAppearance object changes the visual appearance of your data as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

By default, the control's context menu displays the following FormatAppearance objects:

The following screen shot show the control's context menu that contains more FormatAppearance objects


Colors
abc bold
abc italic
abc underline
abc strikeout
abc shadow
abc fine
abc outline
abc gradient
header1
header2
foreground
abc gray
abc red
abc green
abc blue

property Pivot.FormatConditionalAppearances as FormatConditionalAppearances

Retrieves the FormatConditionalAppearances collection of the pivot control.

Type	Description
FormatConditionalAppearances	A FormatConditionalAppearances object that holds a collection of FormatConditionalAppearance objects to be displayed on the control's context menu.

The FormatConditionalAppearances property gives access to a collection of FormatConditionalAppearance objects to be displayed on the control's context menu as seen bellow. Each FormatConditionalAppearance object contains font or color attributes that can be applied to any column/row on the control's list, based on an expression. If the PivotBarVisible property includes the exPivotBarAllowFormatConditionalAppearance flag, the control's context menu includes the FormatConditionalAppearance objects. If the exPivotBarAllowFormatConditionalAppearance flag is missing from the PivotBarVisible property, the control's context menu displays no FormatConditonalAppearance objects. The Add method adds a new FormatConditionalAppearance object to the collection. Use the FormatAppearances property to add support for format the entire column of the pivot table.

The FormatCondtionalAppearance object changes the visual appearance of your data as listed:

- font attributes, like bold, italic,...
- different foreground colors
- different background colors, including the ability to show EBN objects

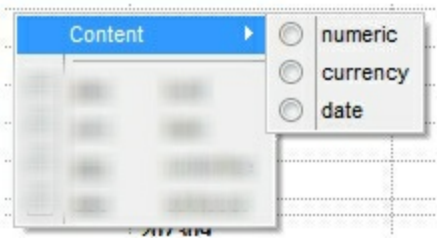
By default, the control's context menu displays the following FormatConditionalAppearance objects:

property Pivot.FormatContents as FormatContents

Retrieves the FormatContents collection of the pivot control.

Type	Description
FormatContents	A FormatContents object that holds a collection of FormatContent object.

The FormatContent object formats or converts the values to be displayed on the columns/rows. For instance, you can specify a column to be display its content as currency, or you can show the control's content in uppercase. The Expression property specifies the format to be applied on objects. The FormatContents collection is displayed on the control's context menu under the Content sub menu as shown bellow. Use the Add method to add new FormatContent objects. If the FormatContent's Expression is not valid, the object shows as disabled in the control's context menu. If the PivotBarVisible property includes the exPivotBarAllowFormatContent flag, the control's context menu includes the FormatContent objects. If the exPivotBarAllowFormatContent flag is missing from the PivotBarVisible property, the control's context menu displays no FormatContent objects.



Here's a few samples on how to use the FormatContent objects:

- *FormatContents.Add("upper","upper(value)")*, displays the column/row in upper-case, such as 'ROMANIA' instead 'Romania'
- *FormatContents.Add("longdate","longdate(date(value))")*, displays the object's content as date in long format, such as 'Monday, December 31, 2012'
- *FormatContents.Add("letter","'<fgcolor=808080>' + upper(value left 1) + '</fgcolor> ' + value")*, shows the first letter twice in bold and gray, such as '**R** Romania' instead 'romania'
- *FormatContents.Add("proper","'' + ((0:=proper(value)) left 1) + '' + (=:0 mid 2)")*, displays the first letter in bold and upper-case, and let the rest unchanged, such as '**M**ihai Filimon' instead 'mihai filimon'.

property Pivot.FormatPivotAggregate as String

Specifies the format to display an aggregate function.

Type	Description
String	A String expression that specifies the format to display the aggregate functions in the control.

By default, the FormatPivotAggregate property is "***proper(caggregate)***". Use the FormatPivotHeader property to determine the way the column displays the captions in the control.

The following keywords are supported by the FormatPivotAggregate:

- **aggregate**, indicates the Key of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection.
- **iaggregate**, indicates the index/position of the Aggregate object associated with the Column, or 0 if no Aggregate function associated. This value indicates the 1-based position of the Aggregate object in the Aggregates collection.
- **naggregate**, indicates the Name of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection.
- **caggregate**, indicates the Caption of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection
- **ilevel**, indicates the index of the level where the Aggregate function is displayed. This is valid only for PivotTotals property, else it is -1.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- ***/*** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

  "expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

  "expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

  If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
  - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

  - 0 - empty (not initialized)
  - 1 - null
  - 2 - short
  - 3 - long
  - 4 - float
  - 5 - double
  - 6 - currency
  - 7 - date
  - 8 - string
  - 9 - object
  - 10 - error
  - 11 - boolean
  - 12 - variant
  - 13 - any
  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long
  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and

Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters

- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

property Pivot.FormatPivotHeader as String

Specifies the format to display the columns in the pivot bar.

Type	Description
String	A String expression that determines the format to show the column's caption in the control's pivot bar.

By default, the FormatPivotHeader property is "***iaggregate ? (caggregate + ' of ' + caption) : caption***", which indicates that the column's default caption is being displayed in no aggregate function is associated with the column, and Caption of the Aggregate object associated with the column, followed by of keyword, and the column's caption, something like Sum of Column 1. Use the FormatPivotHeader property to determine the way the column displays the captions in the control.

The following keywords are supported by the FormatPivotHeader:

- **aggregate**, indicates the Key of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection.
- **iaggregate**, indicates the index/position of the Aggregate object associated with the Column, or 0 if no Aggregate function associated. This value indicates the 1-based position of the Aggregate object in the Aggregates collection.
- **naggregate**, indicates the Name of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection.
- **caggregate**, indicates the Caption of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection.
- **ilevel**, indicates the index of the level where the Aggregate function is displayed. This is valid only for PivotTotals property, else it is -1.
- **caption**, indicates the Caption property of the Column
- **icaption**, indicates the Index property of the Column
- **display**, specifies whether the caption is shown on the control's pivot bar (0), or in the columns header(1)

For instance, "icaption + ' ' + caption" displays the index + caption in the column's header.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- ***/*** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended

using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM,

04:00PM, 06:00PM and 10:00PM
    - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

  - 0 - empty (not initialized)
  - 1 - null
  - 2 - short
  - 3 - long
  - 4 - float
  - 5 - double
  - 6 - currency
  - 7 - date
  - 8 - string
  - 9 - object
  - 10 - error
  - 11 - boolean
  - 12 - variant
  - 13 - any
  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long
  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
  - *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
  - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
    - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
    - 1 - Negative sign, number; for example, -1.1
    - 2 - Negative sign, space, number; for example, - 1.1
    - 3 - Number, negative sign; for example, 1.1-
    - 4 - Number, space, negative sign; for example, 1.1 -

- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)

- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

property Pivot.FormatPivotTotal as String

Specifies the format to display an aggregate/total functions.

Type	Description
String	A String expression that specifies the caption to be displayed for Total/Sub-Total fields.

By default, the FormatPivotTotal property is "***(iaggregate = 5 ? (ilevel = 0 ? 'Total' : 'Subtotal') : caggregate)***". Use the FormatPivotHeader property to determine the way the column displays the captions in the control. Use the FormatPivotAggregate property to determine the way the control displays the aggregate functions.

The following keywords are supported by the FormatPivotTotal:

- **aggregate**, indicates the Key of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection.
- **iaggregate**, indicates the index/position of the Aggregate object associated with the Column, or 0 if no Aggregate function associated. This value indicates the 1-based position of the Aggregate object in the Aggregates collection.
- **naggregate**, indicates the Name of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection.
- **caggregate**, indicates the Caption of the Aggregate object associated with the Column, or empty if the Column belongs to PivotRows collection
- **ilevel**, indicates the index of the level where the Aggregate function is displayed. This is valid only for PivotTotals property, else it is -1.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- **/** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

  "expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

  "expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

  If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
  - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

  - 0 - empty (not initialized)
  - 1 - null
  - 2 - short
  - 3 - long
  - 4 - float
  - 5 - double
  - 6 - currency
  - 7 - date
  - 8 - string
  - 9 - object
  - 10 - error
  - 11 - boolean
  - 12 - variant
  - 13 - any
  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long
  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and

Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters

- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

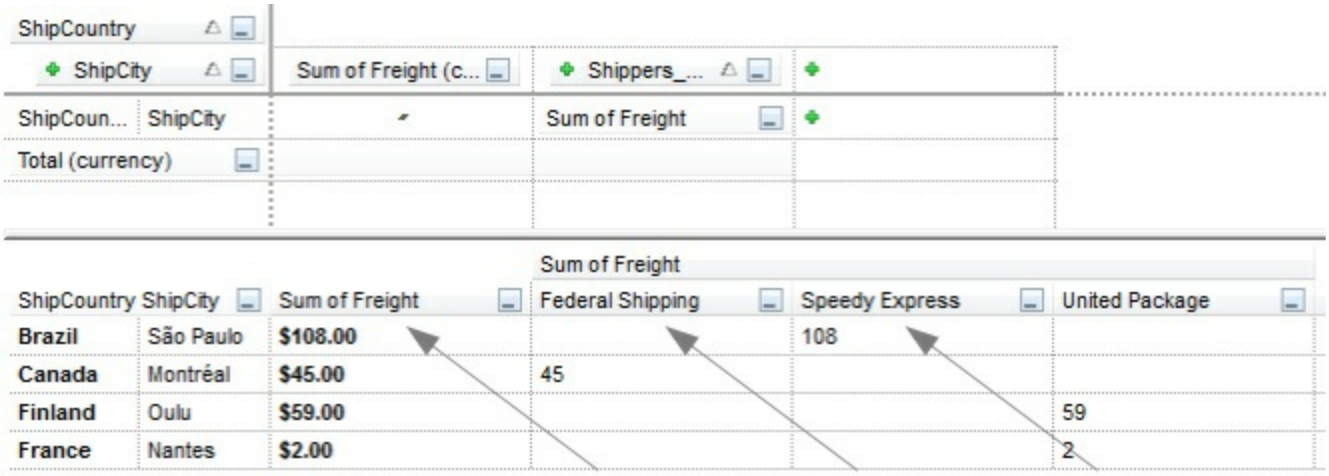
method Pivot.GetHeaders ()

Gets a safe array of all generated columns/headers.

Type	Description
Return	**Description**
Variant	A safe array (two dimensional), that contains the headers of the control's list.

The GetHeaders method returns a two-dimensional safe array that includes all headers being shown on the control's list. The GetItems method gets a two-dimensional safe array that includes all values/items being shown on the control's list.

Having the following layout:



The GetHeaders method gets the following result:

```
-    GetHeaders() as object[,]    {Dimensions:[5, 2]}    object[,]
     [0, 0]   "ShipCountry ShipCity"   object {string}
     [0, 1]   null                     object
     [1, 0]   "Sum of Freight"         object {string}
     [1, 1]   null                     object
     [2, 0]   "Federal Shipping"       object {string}
     [2, 1]   "Sum of Freight"         object {string}
     [3, 0]   "Speedy Express"         object {string}
     [3, 1]   "Sum of Freight"         object {string}
     [4, 0]   "United Package"         object {string}
     [4, 1]   "Sum of Freight"         object {string}
```

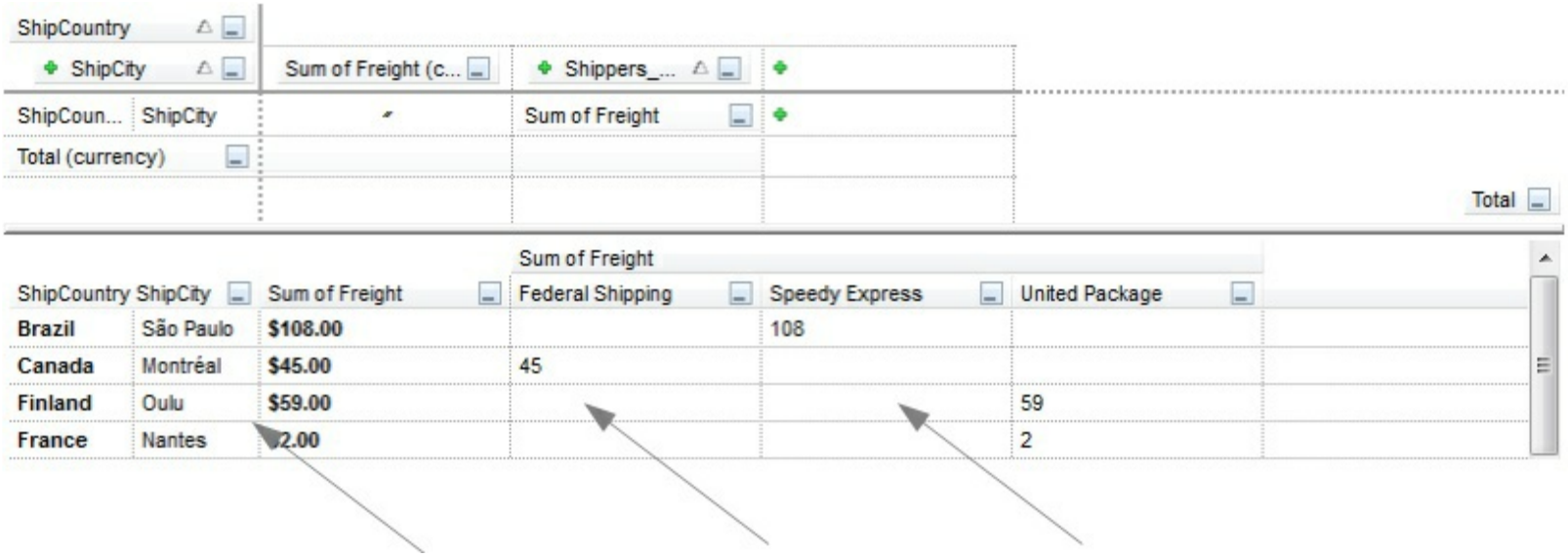
method Pivot.GetItems ()

Gets a safe array of all generated items/values.

Type	Description
Return	**Description**
Variant	A safe array (two dimensional), that contains the values/items of the control's list.

The GetItems method gets a two-dimensional safe array that includes all values/items being shown on the control's list. The GetHeaders method returns a two-dimensional safe array that includes all headers being shown on the control's list.

Having the following layout:



The GetItems method gets the following result:

```
-    GetItems() as object[,]  {Dimensions:[5, 16]}  object[,]
     [0, 0]   "Brazil"    object {string}
     [0, 1]   "Canada"    object {string}
     [0, 2]   "Finland"    object {string}
     [0, 3]   "France"  object {string}
     [0, 4]   "France"  object {string}
     [0, 5]   "France"  object {string}
     [0, 6]   "Germany" object {string}
     [0, 7]   "Germany" object {string}
     [0, 8]   "Germany" object {string}
     [0, 9]   "Ireland" object {string}
     [0, 10] "Mexico"object {string}
```

```
[0, 11] "Poland" object {string}
[0, 12] "Portugal"  object {string}
[0, 13] "Switzerland" object {string}
[0, 14] "USA"  object {string}
[0, 15] "USA"  object {string}
[1, 0]   108.0   object {double}
[1, 1]   45.0 object {double}
[1, 2]   59.0 object {double}
[1, 3]   2.0   object {double}
[1, 4]   30.0 object {double}
[1, 5]   -96.0   object {double}
[1, 6]   77.0 object {double}
[1, 7]   -45.0   object {double}
[1, 8]   1.0   object {double}
[1, 9]   142.0   object {double}
[1, 10] -35.0   object {double}
[1, 11] 81.0 object {double}
[1, 12] -13.0   object {double}
[1, 13] 1.0   object {double}
[1, 14] -147.0 object {double}
[1, 15] -13.0   object {double}
....
```

property Pivot.GridLineColor as Color

Specifies the grid line color.

Type	Description
Color	A color expression that indicates the color of the grid lines.

Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to show the grid lines. The GridLineStyle property to specify the style for horizontal or/and vertical gridlines in the control. Use the LinesAtRoot property specifies whether the control links the root items of the control. Use the HasLines property to specify whether the control draws the link between child items to their corresponding parent item.

property Pivot.GridLineStyle as GridLinesStyleEnum

Specifies the style for gridlines in the list part of the control.

Type	Description
GridLinesStyleEnum	A GridLinesStyleEnum expression that specifies the style to show the control's horizontal or vertical lines.

By default, the GridLineStyle property is exGridLinesDot. The GridLineStyle property has effect only if the DrawGridLines property is not zero. The GridLineStyle property can be used to specify the style for horizontal or/and vertical grid lines. Use the GridLineColor property to specify the color for grid lines. Use the LinesAtRoot property specifies whether the control links the root items of the control. Use the HasLines property to specify whether the control draws the link between child items to their corresponding parent item.

The following VB sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = GridLinesStyleEnum.exGridLinesHDash Or
GridLinesStyleEnum.exGridLinesVSolid
```

The following VB/NET sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = exontrol.EXGRIDLib.GridLinesStyleEnum.exGridLinesHDash Or
exontrol.EXGRIDLib.GridLinesStyleEnum.exGridLinesVSolid
```

The following C# sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = exontrol.EXGRIDLib.GridLinesStyleEnum.exGridLinesHDash |
exontrol.EXGRIDLib.GridLinesStyleEnum.exGridLinesVSolid;
```

The following Delphi sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle := Integer(EXGRIDLib.GridLinesStyleEnum.exGridLinesHDash) Or
Integer(EXGRIDLib.GridLinesStyleEnum.exGridLinesVSolid);
```

The following VFP sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = 36
```

property Pivot.HasLines as HierarchyLineEnum

Enhances the graphic representation of a grid control's hierarchy by drawing lines that link child items to their corresponding parent item.

Type	Description
HierarchyLineEnum	An HierarchyLinesEnum expression that indicates whether the control displays the hierarchy lines.

Use the HasLines property to hide the hierarchy lines. Use the LinesAtRoot property to allow control displays a line that links that root items of the control. Use the DrawGridLines property to display grid lines. The GridLineStyle property to specify the style for horizontal or/and vertical gridlines in the control.

property Pivot.HeaderAppearance as AppearanceEnum

Retrieves or sets a value that indicates the header's appearance.

Type	Description
AppearanceEnum	A AppearanceEnum expression that specifies the header's appearance.

The HeaderAppearance property retrieves or sets a value that indicates the header's appearance.

property Pivot.HeaderHeight as Long

Retrieves or sets a value indicating the control's header height.

Type	Description
Long	A long expression that indicates the height of the control's header bar.

By default, the HeaderHeight property is 18 pixels. Use the HeaderHeight property to change the height of the control's header bar. Use the HeaderVisible property to hide the control's header bar. The HeaderHeight property specifies the height of the control's header, and so the height of the columns to be displayed on the control's pivot bar. Use the DefaultColumnWidth property to specify the default for the width of the columns.

property Pivot.HeaderVisible as Boolean

Retrieves or sets a value that indicates whether the the control's header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the columns header bar is visible or hidden.

Use the HeaderVisible property to hide the columns header bar. Use the HeaderHeight property to specify the height of the control's header bar. Use the BackColorHeader property to specify the header's background color. The Background(exCursorHoverColumn) property specifies the visual appearance of the column's header when the cursor hovers it. Use the DefaultColumnWidth property to specify the default for the width of the columns.

property Pivot.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

Type	Description
Key as String	A String expression that indicates the key of the picture being added or replaced. If the Key property is Empty string, the entire collection of pictures is cleared.
Variant	The HTMLPicture specifies the picture being associated to a key. It can be one of the followings: • a string expression that indicates the path to the picture file, being loaded. • a string expression that indicates the base64 encoded string that holds a picture object, Use the eximages tool to save your picture as base64 encoded format. • A Picture object that indicates the picture being added or replaced. (A Picture object implements IPicture interface), If empty, the picture being associated to a key is removed. If the key already exists the new picture is replaced. If the key is not empty, and it doesn't not exist a new picture is added

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> tags. By default, the HTMLPicture collection is empty. Use the HTMLPicture property to add new pictures to be used in HTML captions. For instance, the HTMLPicture("pic1") = "c:\winnt\zapotec.bmp", loads the zapotec picture and associates the pic1 key to it. Any "<img>pic1</img>" sequence in HTML captions, displays the pic1 picture. On return, the HTMLPicture property retrieves a Picture object (this implements the IPictureDisp interface).

property Pivot.hWnd as Long

Retrieves the control's window handle.

Type	Description
Long	A long expression that indicates the control's window handle.

Use the hWnd property to get the control's main window handle. The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

method Pivot.Images (Handle as Variant)

Sets the control's image list at runtime.

Type	Description
Handle as Variant	The Handle parameter can be: • A string expression that specifies the ICO file to add. The ICO file format is an image file format for computer icons in Microsoft Windows. ICO files contain one or more small images at multiple sizes and color depths, such that they may be scaled appropriately. For instance, Images("c:\temp\copy.ico") method adds the sync.ico file to the control's Images collection *(string, loads the icon using its path)* • A string expression that indicates the BASE64 encoded string that holds the icons list. Use the Exontrol's ExImages tool to save/load your icons as BASE64 encoded format. In this case the string may begin with "gBJJ..." *(string, loads icons using base64 encoded string)* • A reference to a Microsoft ImageList control (mscomctl.ocx, MSComctlLib.ImageList type) that holds the icons to add *(object, loads icons from a Microsoft ImageList control)* • A reference to a Picture (IPictureDisp implementation) that holds the icon to add. For instance, the VB's LoadPicture (Function LoadPicture([FileName], [Size], [ColorDepth], [X], [Y]) As IPictureDisp) or LoadResPicture (Function LoadResPicture(id, restype As Integer) As IPictureDisp) returns a picture object (object, loads icon from a Picture object) • A long expression that identifies a handle to an Image List Control (the Handle should be of HIMAGELIST type). On 64-bit platforms, the Handle parameter must be a Variant of LongLong / LONG_PTR data type (signed 64-bit (8-byte) integers), saved under llVal field, as VT_I8 type. The LONGLONG / LONG_PTR is __int64, a 64-bit integer. For instance, in C++ you can use as Images(COleVariant((LONG_PTR)hImageList)) or Images(COleVariant((LONGLONG)hImageList)), where hImageList is of

HIMAGELIST type. The GetSafeHandle() method of the CImageList gets the HIMAGELIST handle (long, loads icon from HIMAGELIST type)

The user can add images at design time, by drag and drop files to combo's image holder (ShowImageList property). The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. Use the CheckImage or RadioImage property to specify a different look for checkboxes or radio buttons in the cells. The FormatImage property defines the expression to determine the images the column display.

property Pivot.ImageSize as Long

Retrieves or sets the size of control' icons/images/check-boxes/radio-buttons.

Type	Description
Long	A long expression that defines the size of icons the control displays.

By default, the ImageSize property is 16 (pixels). The ImageSize property specifies the size of icons being loaded using the Images method. The control's Images collection is cleared if the ImageSize property is changed, so it is recommended to set the ImageSize property before calling the Images method. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. For instance, if the ICO file to load includes different types the one closest with the size specified by ImageSize property is loaded by Images method. The ImageSize property does NOT change the height for the control's font.

The ImageSize property defines the size to display the following UI elements:

- any icon that a cell or column displays
- check-box or radio-buttons
- expand/collapse glyphs
- header's sorting or drop down-filter glyphs

method Pivot.Import ([Source as Variant], [Options as Variant])

Imports the control's data from a CSV format.

Type	Description
Source as Variant	The Source parameter may be one of the following: • A String expression that indicates the path to a file to be loaded or the content itself (If the expression points to a file, the file's content is loaded) • a Safe Array one-dimension or two dimensional to be loaded. If the Source parameter points to a one-dimension safe array, it indicates the rows to be loaded. If the Source parameter refers a two-dimension safe array, the first dimension indicates the rows, while the second indicates the column. If the Options parameter includes the word "reverse", the first dimension indicates the columns, while the second indicates the rows of data.
Options as Variant	A String expression that specifies different options to be used when loading data using the Import method as explained bellow.

Return	Description
Variant	Reserved for future use only.

The Import method loads CSV files / CSV content or data from a safe-array as explained:

- A String expression that indicates the path to a file to be loaded or the content itself. If the expression points to a file, the file's content is loaded.The Import method loads CSV files or CSV content. A comma-separated values (CSV) file stores tabular data (numbers and text) in plain-text form. Plain text means that the file is a sequence of characters, with no data that has to be interpreted instead, as binary numbers. A CSV file consists of any number of records, separated by line breaks of some kind; each record consists of fields, separated by some other character or string, most commonly a literal comma or tab. Usually, all records have an identical sequence of fields. The Source parameter indicates the path to the CSV file to be loaded, or the content itself. For instance, the *Import "C:\Program Files\Exontrol\ExPivot\Sample\data.txt"* imports the data.txt file.
- A Safe array of one-dimensional or two-dimension type, which indicates the data to be loaded. The safe array must be with 1 or 2 dimensions, else an error occurs. If the Source parameter refers a one-dimensional array, the content of the array indicates

the values of the rows. If the Source parameter refers a two-dimensional array, the first dimension indicates the columns, while the second indicates the rows (unless the Options parameter indicates the "reverse" word, which reverses the columns with rows . For instance, in VB6 you can load data from a safe array using the syntax *Import Array(1, 1, 2, 2, 3, 3, 4, 4)*. Where Array is a method of VB6 that creates a safe array of different values. The same way you can have the declaration *Dim data(1,99) as String*, and call the *Import data*, will load a two-dimensional array. The sample loads 2 columns, and 100 rows from the data array.

Here's a few samples of using Import method with arrays, using the /COM version:

- **VB,VBA**: .Import (Array(1, 2, 3, 4, 5))
- **VB/NET**: .GetOcx().Import(New Integer() {1, 2, 3, 4, 5})
- **C#**: .Import(new int[] { 1, 2, 3, 4, 5 }, null);

Here's a few samples of using Import method with arrays, using the /NET version:

- **VB/NET**: .Import(New Integer() {1, 2, 3, 4, 5})
- **C#**: .Import(new int[] { 1, 2, 3, 4, 5 });;

The DataColumns property accesses the control's Columns collection, so you can rename or specify the column's type once the control's data is loaded. The PivotColumnsFloatBarVisible property specifies whether the Columns collection is displayed to a floating bar, so user can drag and drop columns to the control's pivot bar so it gets data summarized. The ClearData method clears the control's data. Use the DisplayPivotData property to specify the number of rows to be displayed on the control's list. The LoadHeadersOnly property loads the headers only, so no data is loaded.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method
- The user can drag and drop any TXT or XML files to the control. Use the AllowDrop property on False, to prevent loading the data-files (TXT, XML files), by drag and drop, into the control.

The AppendData method appends data to the control (prevents clearing data already loaded).

The following properties may be used to group and summarize the data, once it is loaded:

- PivotRows property specifies the list of DATA columns that determines the first column in the control's list. In other words, the Group-By columns

- PivotColumns property specifies the list of DATA columns that determines the rest of the columns to be displayed on the control's list
- PivotTotals property specifies the list of total/sub-total functions to be displayed on the control's list.

If the Source parameter points to a string, the Options parameter indicates a list of options separated by space character (or \r, \n or \r\n newline delimiters) as listed option=value, where the option and the value could be:

- **eor**, (End Of Row), specifies the delimiter of the row data, \r\n (new-line separator) if missing. You can use the ' or ` to specify the value as a string.
- **eof,** (End of Field), indicates the delimiter of fields inside the row, \t (tab character) if missing. You can use the ' or ` to specify the value as a string.
- **str**, (STRing), specifies the characters to delimit the strings in the row data, nothing if missing. You can use the ' or ` to specify the value as a string.
- **hdr**, (HeaDR), specifies if the data reads the first line as a header to be imported. Could be 1 or 0.

For instance, if the Options parameter is "hdr=0 eof=';'", it indicates that the data in the Source contains no header information, and the fields inside the data are separated by a ; character. If the Options parameter is "eof=';' str=`'`" it indicates that the separator of the fields in the data is ; character while the ' character delimits the strings inside the data.

If the Source parameter points to a safe array (two-dimensional), the Options parameter may include the following word:

- **reverse**, the first dimension of the array indicates the columns, and the second dimension specifies the rows of data. If the reverse word is missing, the first dimension indicates the rows, while the second dimension specifies the columns.

The following samples show how to load data from specified text (item 1;item 2#item 3;item 4), using different options (eor='#' eof=';' hdr=0). The data looks as follows once the Import method is called

Column 1	Column 2
item 1	item 2
item 3	item 4

VB6, VBA (MS Access, Excell...)

```
With Pivot1
   .Import "item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0"
End With
```

VB.NET

```
With Expivot1
   .Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0")
End With
```

VB.NET for /COM

```
With AxPivot1
   .Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0")
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXPIVOTLib' for the library: 'ExPivot 1.0 Control Library'

   #import <ExPivot.dll>
   using namespace EXPIVOTLib;
*/
EXPIVOTLib::IPivotPtr spPivot1 = GetDlgItem(IDC_PIVOT1)->GetControlUnknown();
spPivot1->Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0");
```

C++ Builder

```
Pivot1->Import(TVariant("item 1;item 2#item 3;item 4"),TVariant("eor='#' eof=';' hdr=0"));
```

C#

```
expivot1.Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0");
```

JavaScript

```
<OBJECT classid="clsid:5C9DF3D3-81B1-42C4-BED6-658F17748686" id="Pivot1">
</OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
   Pivot1.Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0");
</SCRIPT>
```

C# for /COM

```
axPivot1.Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0");
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   ;

   super();

   expivot1.Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0");
}
```

Delphi 8 (.NET only)

```
with AxPivot1 do
begin
   Import('item 1;item 2#item 3;item 4','eor=''#'' eof=';'' hdr=0');
end
```

Delphi (standard)

```
with Pivot1 do
begin
   Import('item 1;item 2#item 3;item 4','eor=''#'' eof=';'' hdr=0');
end
```

VFP

```
with thisform.Pivot1
   .Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0")
```

```
endwith
```

dBASE Plus

```
local oPivot

oPivot = form.Activex1.nativeObject
oPivot.Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0")
```

XBasic (Alpha Five)

```
Dim oPivot as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
oPivot.Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0")
```

Visual Objects

```

oDCOCX_Exontrol1:Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0")
```

PowerBuilder

```
OleObject oPivot

oPivot = ole_1.Object
oPivot.Import("item 1;item 2#item 3;item 4","eor='#' eof=';' hdr=0")
```

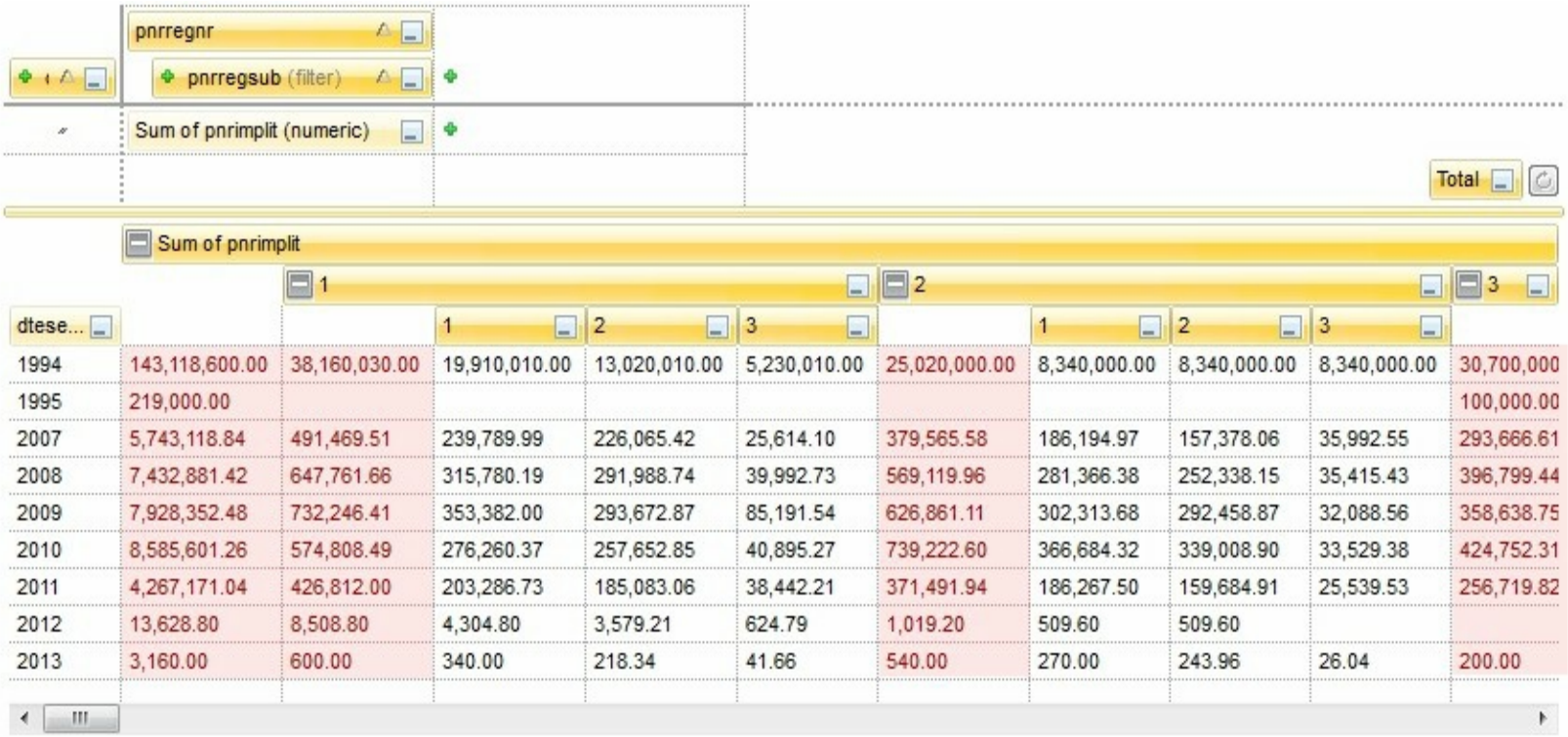
property Pivot.IncludeExpandColumn as IncludeExpandColumnEnum

Specifies whether the column itself is displayed in the list (header/chart), while it expanded (the column contains child columns).

Type	Description
IncludeExpandColumnEnum	An IncludeExpandColumnEnum expression that specifies whether the column itself is displayed in the list (header/chart), while it expanded (the column contains child columns).

By default, the IncludeExpandColumn property is exExcludeExpandColumn (0), which indicates that the column's data is not shown on the chart when it contains child columns, and it is shown as expanded. For instance, you can use the IncludeExpandColumn property to show totals on rows.

The following screen shot shows the control, when IncludeExpandColumn property is exIncludeExpandColumn:



dtese...		1	1	2	3	2	1	2	3	3
1994	143,118,600.00	38,160,030.00	19,910,010.00	13,020,010.00	5,230,010.00	25,020,000.00	8,340,000.00	8,340,000.00	8,340,000.00	30,700,000
1995	219,000.00									100,000.00
2007	5,743,118.84	491,469.51	239,789.99	226,065.42	25,614.10	379,565.58	186,194.97	157,378.06	35,992.55	293,666.61
2008	7,432,881.42	647,761.66	315,780.19	291,988.74	39,992.73	569,119.96	281,366.38	252,338.15	35,415.43	396,799.44
2009	7,928,352.48	732,246.41	353,382.00	293,672.87	85,191.54	626,861.11	302,313.68	292,458.87	32,088.56	358,638.75
2010	8,585,601.26	574,808.49	276,260.37	257,652.85	40,895.27	739,222.60	366,684.32	339,008.90	33,529.38	424,752.31
2011	4,267,171.04	426,812.00	203,286.73	185,083.06	38,442.21	371,491.94	186,267.50	159,684.91	25,539.53	256,719.82
2012	13,628.80	8,508.80	4,304.80	3,579.21	624.79	1,019.20	509.60	509.60		
2013	3,160.00	600.00	340.00	218.34	41.66	540.00	270.00	243.96	26.04	200.00

The following screen shot shows the control, when IncludeExpandColumn property is exExcludeExpandColumn (default):

pnrregnr

pnrregsub (filter)

Sum of pnrimplit (numeric)

Total

	Sum of pnrimplit									
	1			2			3			4
dtese...	1	2	3	1	2	3	1	2	3	1
1994	19,910,010.00	13,020,010.00	5,230,010.00	8,340,000.00	8,340,000.00	8,340,000.00	16,680,000.00	10,020,000.00	4,000,000.00	8,404,260.00
1995								100,000.00		
2007	239,789.99	226,065.42	25,614.10	186,194.97	157,378.06	35,992.55	147,625.89	130,178.74	15,861.98	131,285.65
2008	315,780.19	291,988.74	39,992.73	281,366.38	252,338.15	35,415.43	183,170.36	165,507.45	48,121.63	147,605.99
2009	353,382.00	293,672.87	85,191.54	302,313.68	292,458.87	32,088.56	185,284.07	151,126.13	22,228.55	176,897.28
2010	276,260.37	257,652.85	40,895.27	366,684.32	339,008.90	33,529.38	211,734.30	192,762.52	20,255.49	134,503.92
2011	203,286.73	185,083.06	38,442.21	186,267.50	159,684.91	25,539.53	130,645.61	117,969.49	8,104.72	93,186.77
2012	4,304.80	3,579.21	624.79	509.60	509.60					
2013	340.00	218.34	41.66	270.00	243.96	26.04	100.00	82.64	17.36	100.00

property Pivot.Indent as Long

Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.

Type	Description
Long	A long expression that indicates the amount, in pixels, that child items are indented relative to their parent items.

If the Indent property is 0, the child items are not indented relative to their parent item. Use HasLines and LinesAtRoot properties to show the hierarchy lines.

property Pivot.Layout as String

Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.

Type	Description
String	A String expression that specifies the control's layout.

You can use the Layout property to store the control's layout and to restore the layout later. For instance, you can save the control's Layout property to a file when the application is closing, and you can restore the control's layout when the application is loaded. The Layout property saves almost all of the control's properties that user can change at runtime (like changing the column's position by drag and drop). The Layout property does NOT save the control's data, so the Layout property should be called once you loaded the data from your database, xml or any other alternative. Once the data is loaded, you can call the Layout property to restore the View as it was saved. Before closing the application, you can call the Layout property and save the content to a file for reading next time the application is opened.

The Layout property saves/loads the following information:

- PivotRows property
- PivotColumns property,
- PivotTotals property,
- columns size and position,
- current selection,
- scrolling position and size,
- expanded/collapsed items, if any
- sorting columns,
- filtering options

These properties are serialized to a string and encoded in BASE64 format.

The following movies show how Layout works:

- The Layout property is used to save and restore the control's view.

property Pivot.LinesAtRoot as LinesAtRootEnum

Link items at the root of the hierarchy.

Type	Description
LinesAtRootEnum	A LinesAtRootEnum expression that indicates whether the control links the items at the root of the hierarchy.

Use the Indent property to increase or decrease the amount, in pixels, that child items are indented relative to their parent items. Use the HasLines property to enhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their corresponding parent item.

property Pivot.LoadHeadersOnly as Boolean

Loads the headers only, so no data is loaded.

Type	Description
Boolean	A Boolean expression that specifies whether the control loads the headers only.

By default, the LoadHeadersOnly property is False, which indicates that headers and data is loaded by Import or DataSource property. If the LoadHeadersOnly property is True, the Import or DataSource property loads just the headers of the columns. The LoadHeadersOnly property loads the headers only, so no data is loaded. The new value for the LoadHeadersOnly property has effect for the next calling of Import or DataSource property.

method Pivot.LoadXML (Source as Variant)

Loads an XML document from the specified location, using MSXML parser.

Type	Description
Source as Variant	An indicator of the object that specifies the source for the XML document. The object can represent a file name, a URL, an IStream, a SAFEARRAY, or an IXMLDOMDocument

Return	Description
Boolean	A boolean expression that specifies whether the XML document is loaded without errors. If an error occurs, the method retrieves a description of the error occurred.

The LoadXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to load XML documents, previously saved using the SaveXML method. The control is emptied when the LoadXML method is called, and so the columns and items collection are emptied before loading the XML document. The DataColumns property accesses the control's Columns collection, so you can rename or specify the column's type once the control's data is loaded. The PivotColumnsFloatBarVisible property specifies whether the Columns collection is displayed to a floating bar, so user can drag and drop columns to the control's pivot bar so it gets data summarized. The ClearData method clears the control's data. Use the DisplayPivotData property to specify the number of rows to be displayed on the control's list.

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method
- The user can drag and drop any TXT or XML files to the control.

The AppendData method appends data to the control (prevents clearing data already loaded).

The following properties may be used to group and summarize the data, once it is loaded:

- PivotRows property specifies the list of DATA columns that determines the first column in the control's list. In other words, the Group-By columns
- PivotColumns property specifies the list of DATA columns that determines the rest of the columns to be displayed on the control's list
- PivotTotals property specifies the list of total/sub-total functions to be displayed on the control's list.

property Pivot.LockRowsColumn as Boolean

Retrieves or sets a value that indicates whether the rows column in the list is locked or scrollable.

Type	Description
Boolean	A Boolean expression that specifies whether the first column is locked or unlocked.

By default, LockRowsColumn property property is True. The LockRowsColumn property specifies whether the first column is locked or unlocked. By lock a row or a column it means that the position of it remains unchanged when the user scrolls the control's content. The PivotRows property specifies the list of columns that builds the first column in the control's list. The LockTotalRows property specifies whether the grand total rows are locked or unlocked. The PivotTotals property specifies the total/aggregate rows. The SelectableAggregateRows property specifies whether the inside total/aggregate rows are selectable or unselectable.

property Pivot.LockTotalRows as Boolean

Retrieves or sets a value that indicates whether the total rows in the list are locked or scrollable.

Type	Description
Boolean	A Boolean expression that specifies whether the grand total rows are locked or unlocked.

By default, the LockTotalRows property is True. The LockTotalRows property specifies whether the grand total rows are locked or unlocked. By lock a row or a column it means that the position of it remains unchanged when the user scrolls the control's content. The PivotTotals property specifies the total/aggregate rows. The LockRowsColumn property specifies whether the first column is locked or unlocked. The SelectableAggregateRows property specifies whether the inside total/aggregate rows are selectable or unselectable.

property Pivot.OnFilterChange as OnFilterChangeEnum

Specifies the action that the control performs once the user changes the filter at runtime.

Type	Description
OnFilterChangeEnum	An OnFilterChangeEnum expression that specifies what's happen once the user applies a filter.

By default, the OnFilterChange property is exFilterUpdateTotals. In other words, the total/subtotal fields are updated so it gets data only for visible rows, not for the entire data. Use the OnFilterChange property to hide the total/sub-total fields when a filter is applied.

The OnFilterChange property can do one of the following:

- updates the total/sub-total fields, so it gets the value for the visible rows only, not for the entire data
- hide the total/sub-total fields, once the user applies the filter

property Pivot.PaneHeight(Bottom as Boolean) as Long

Specifies the height for the top or bottom panel.

Type	Description
Bottom as Boolean	A Boolean expression that specifies part to be changed.
Long	A Long expression that specifies the height of the panel.

Use the PaneHeight property to specify the height on the top or bottom. The top panel is the control's pivot bar, while the bottom part is the control's list where the result goes. If the exPivotBarSizable is present in the PivotBarVisible property, the user can resize the pivot bar by dragging the bottom side of the control. The resize cursor is shown when the pivot bar is resizable and cursor hovers the bottom side of the pivot bar. The PaneMinHeight property specifies the minimum height of the top/bottom parts.

property Pivot.PaneMinHeight(Bottom as Boolean) as Long

Specifies the minimum height for the top or bottom panel.

Type	Description
Bottom as Boolean	A Boolean expression that specifies part to be changed.
Long	A Long expression that specifies the minimum height of the panel.

The PaneMinHeight property specifies the minimum height of the top/bottom parts. Use the PaneHeight property to specify the height on the top or bottom. The top panel is the control's pivot bar, while the bottom part is the control's list where the result goes. If the exPivotBarSizable is present in the PivotBarVisible property, the user can resize the pivot bar by dragging the bottom side of the control. The resize cursor is shown when the pivot bar is resizable and cursor hovers the bottom side of the pivot bar.

property Pivot.Picture as IPictureDisp

Retrieves or sets a graphic to be displayed in the control.

Type	Description
IPictureDisp	A Picture object that's displayed on the control's background.

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the BackColor property to specify the control's background color.

property Pivot.PictureDisplay as PictureDisplayEnum

Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way how the picture is displayed.

By default, the PictureDisplay property is exTile. The PictureDisplay property specifies how the Picture is displayed on the control's background. If the control has no picture associated the PictureDisplay property has no effect. Use the BackColor property to specify the control's background color.

property Pivot.PivotBarVisible as PivotBarVisibleEnum

Specifies how the control displays its pivot bar.

Type	Description
PivotBarVisibleEnum	A PivotBarVisibleEnum expression that specifies the properties of the control's pivot bar. This is a bit-OR combination on the listed values.

By default, the PivotBarVisible property is exPivotBarDefault, which is a BIT-OR combination of the following values. exPivotBarVisible | exPivotBarSizable | exPivotBarAutoFit | exPivotBarShowTotals | exPivotBarAllowValues | exPivotBarAllowFormatConditionalAppearance | exPivotBarAllowFormatAppearance | exPivotBarAllowFormatContent | exPivotBarAutoUpdate | exPivotBarAllowUndoRedo | exPivotBarAllowResizeColumns. The PivotColumnsFloatBarVisible property indicates whether the control shows a floating panel to display the pivot columns that can be dropped to the control's pivot bar. The LayoutStartChanging/LayoutEndChanging events notify your application once changes occurs in the control.

The following screen shot shows the control's pivot bar:

Use the PivotBarVisible property to specify one or more of the followings:

- shows or hides the control's pivot bar
- allow resizing the control's pivot bar
- show the control's pivot bar to a floating panel
- auto-resize the control's pivot bar so its content fits the client area
- show or hides the total aggregate functions
- enables or disables the Auto-Hide feature, so the pivot bar is shown only when the cursor hovers the pivot bar
- allows or prevents displaying value columns
- prevent showing the appearances or format content to the control's context menu
- shows or hides the Refresh button, in case the grouping of data is time consuming
- enables or disables the control's Undo/Redo feature
- allows or prevents resizing the columns in the pivot bar

property Pivot.PivotColumns as String

Specifies the list of columns to be displayed in the list.

Type	Description
String	A String expression that indicates the list of data columns that shows the summarized data. For instance, the "sum(5),sum(5)/12" display the SUM for the column with the index 5, and adds a new column for each value found in the column with the index 12, by displaying the SUM of column with the index 5.

The PivotColumns property specifies the list of columns that shows the summarized data. The PivotColumns property has effect only if it is valid and the PivotRows property is set. The DisplayPivotFields property specifies the number of maximum columns to be added during the execution of the current layout. Use the Layout property to save or restore the control's view once the user closes/runs the application. The Aggregates collection holds a collection of Aggregate functions the user can display to summarize the data. Use the FormatPivotAggregate / FormatPivotTotal property to display aggregate functions in a different format. The FormatConditionalAppearances helps you to provide conditional-format for your data, or in other words, ability to highlight values that matches a specified expression.

For instance:

- *"sum(5)[bold,content=numeric]"*, adds a single column that displays the sum of data column with the index 5, and its content will be displayed as numeric and in bold.
- *"sum(5)[content=numeric]/12"*, adds a column for each unique value found in the data column with the index 12, and each column displays the sum of data column with the index 5 as numeric, associated with the value of the data column.
- *"sum(5)[content=numeric]/12;6"*, groups by data columns with the index 12 and 6, and adds a new column for each unique value found, by displaying the sum of data column with the index 5 as numeric, for the associated values in the data columns.
- *"sum(5)[bold,content=numeric],sum(5)[content=numeric]/12;6"*, adds a single column that displays the sum of data column with the index 5 in bold as numeric, and follows the group-by data columns shown earlier.

The PivotColumns in **BNF** notation is:

```
PivotColumns ::= "<Aggregate>[,<Aggregate>]"
Aggregate ::= <AggregateKey>(<Index>)[<Options>] | <AggregateKey>(<Index>)
[<Options>]/<Index>[:<Order>][<Options>][;<Index>[:<Order>][<Options>]]
Index ::= <Digit>[<Digit>]
```

```
Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Order ::= A | D
Options ::= \[<Option> | ,<Option>\]
Option ::= <CheckOption> | content=<RadioOption>
```

where

- AggregateKey, is any Key in the Aggregates collection
- Index, is the index of the Column, a value between 0 and DataColumns.Count - 1
- CheckOption is any Key in the FormatAppearances collection
- RadioOption is any Key in the FormatContents collection

The following screen shot shows how the columns in the control's list, may look like if the PivotColumns property is "*sum(5)[bold,content=numeric]*":

ShipCountry	Σ of Freight
Argentina	1,777.00
Austria	27,730.00
Belgium	4,274.00
Brazil	14,586.00
Canada	6,324.00
Denmark	4,283.00
Finland	2,873.00
France	12,627.00
Germany	38,171.00
Ireland	7,214.00
Italy	2,106.00
Mexico	3,227.00
Norway	898.00
Poland	462.00
Portugal	1,747.00
Spain	2,416.00
Sweden	10,141.00
Switzerland	3,911.00
UK	8,486.00
USA	46,181.00
Venezuela	7,875.00

The following screen shot shows how the columns in the control's list, may look like if the PivotColumns property is "*sum(5)[content=numeric]/12*":

ShipCountry	Shippers_Co...	
	Σ of Freight (num...	

	Σ of Freight		
ShipCountry	United Package	Speedy Express	Federal Shipping
Argentina	1,413.00	253.00	111.00
Austria	12,100.00	9,065.00	6,565.00
Belgium	2,410.00	765.00	1,099.00
Brazil	7,536.00	4,823.00	2,227.00
Canada	3,448.00	725.00	2,151.00
Denmark	2,027.00	883.00	1,373.00
Finland	1,667.00	657.00	549.00
France	3,653.00	2,908.00	6,066.00
Germany	13,579.00	12,763.00	11,829.00
Ireland	4,580.00	793.00	1,841.00
Italy	410.00	1,195.00	501.00
Mexico	1,508.00	497.00	1,222.00
Norway	833.00	26.00	39.00
Poland	323.00	9.00	130.00
Portugal	1,017.00	134.00	596.00
Spain	1,612.00	517.00	287.00
Sweden	5,383.00	3,159.00	1,599.00
Switzerland	1,045.00	766.00	2,100.00
UK	3,501.00	2,434.00	2,551.00
USA	20,041.00	7,768.00	18,372.00
Venezuela	3,501.00	2,144.00	2,230.00

The following screen shot shows how the columns in the control's list, may look like if the PivotColumns property is "*sum(5)[content=numeric]/12;6*":

ShipCountry | Shippers_Company... | Suppliers_Co... | Σ of Freight (num...

	Σ of Freight					
	United Package	Speedy Express	Federal Shipping			
ShipCountry			Exotic Liquids	Refrescos America...	Bigfoot Breweries	Aux joyeux ecclésia...
Argentina	1,413.00	253.00			32.00	
Austria	12,100.00	9,065.00	145.00	256.00	197.00	506.00
Belgium	2,410.00	765.00	6.00	6.00		
Brazil	7,536.00	4,823.00	99.00	77.00	27.00	3.00
Canada	3,448.00	725.00	338.00	1.00		
Denmark	2,027.00	883.00		145.00	14.00	70.00
Finland	1,667.00	657.00	29.00			
France	3,653.00	2,908.00	30.00	40.00	263.00	132.00
Germany	13,579.00	12,763.00	1,008.00	31.00	229.00	1,010.00
Ireland	4,580.00	793.00	142.00			18.00
Italy	410.00	1,195.00				70.00
Mexico	1,508.00	497.00	73.00	21.00	85.00	
Norway	833.00	26.00				
Poland	323.00	9.00	24.00		12.00	
Portugal	1,017.00	134.00		1.00	73.00	
Spain	1,612.00	517.00				
Sweden	5,383.00	3,159.00	4.00		4.00	208.00
Switzerland	1,045.00	766.00	148.00			
UK	3,501.00	2,434.00	82.00		114.00	22.00
USA	20,041.00	7,768.00	713.00	519.00	326.00	2,091.00
Venezuela	3,501.00	2,144.00	200.00		105.00	83.00

The following screen shot shows how the columns in the control's list, may look like if the PivotColumns property is "*sum(5)[bold,content=numeric],sum(5)[content=numeric]/12;6*":

ShipCountry	Σ of Freight (n...	Shippers_Company...	
		Suppliers_Co...	
		Σ of Freight (num...	

		Σ of Freight				
		United Package	Speedy Express	Federal Shipping		
ShipCountry	Σ of Freight			Exotic Liquids	Refrescos America...	Bigfoot Breweries
Argentina	1,777.00	1,413.00	253.00			32.00
Austria	27,730.00	12,100.00	9,065.00	145.00	256.00	197.00
Belgium	4,274.00	2,410.00	765.00	6.00	6.00	
Brazil	14,586.00	7,536.00	4,823.00	99.00	77.00	27.00
Canada	6,324.00	3,448.00	725.00	338.00	1.00	
Denmark	4,283.00	2,027.00	883.00		145.00	14.00
Finland	2,873.00	1,667.00	657.00	29.00		
France	12,627.00	3,653.00	2,908.00	30.00	40.00	263.00
Germany	38,171.00	13,579.00	12,763.00	1,008.00	31.00	229.00
Ireland	7,214.00	4,580.00	793.00	142.00		
Italy	2,106.00	410.00	1,195.00			
Mexico	3,227.00	1,508.00	497.00	73.00	21.00	85.00
Norway	898.00	833.00	26.00			
Poland	462.00	323.00	9.00	24.00		12.00
Portugal	1,747.00	1,017.00	134.00		1.00	73.00
Spain	2,416.00	1,612.00	517.00			
Sweden	10,141.00	5,383.00	3,159.00	4.00		4.00
Switzerland	3,911.00	1,045.00	766.00	148.00		
UK	8,486.00	3,501.00	2,434.00	82.00		114.00
USA	46,181.00	20,041.00	7,768.00	713.00	519.00	326.00
Venezuela	7,875.00	3,501.00	2,144.00	200.00		105.00

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method
- The user can drag and drop any TXT or XML files to the control.

Once the data is loaded to the control, the user can drag and drop columns to summarize the data or set the following properties in the following order:

- PivotRows property, specifies the list of columns that determines the first column in the control's list
- PivotColumns property, specifies the list of columns that determines the rest of the columns in the control's list
- PivotTotals property, specifies the list of aggregate functions to be displayed in the control's list

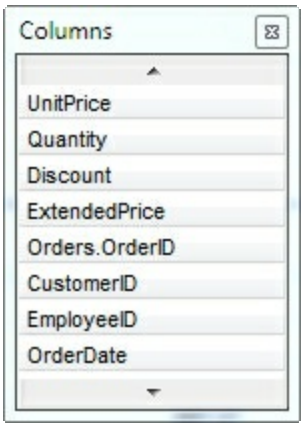
property Pivot.PivotColumnsFloatBarVisible as Boolean

Retrieves or sets a value that indicates whether the pivot columns float bar is visible or hidden.

Type	Description
Boolean	A Boolean expression that specifies whether the columns floating bar is shown or hidden.

By default, the PivotColumnsFloatBarVisible property is False. Use the PivotColumnsFloatBarVisible property to display a floating window that shows the data columns to be summarized by drag and drop to the control's pivot bar. The PivotBarVisible property indicates whether the control's pivot bar is visible or hidden. The user drag and drop columns from this panel to summarize the data. Use the DataColumns property to access the data columns collection once data was provided using the Import, DataSource or LoadXML methods.

The following screen shot shows the control's Columns to a floating bar:



Use the following properties to customize the visual appearance of the control's Columns Floating Panel:

- Background(exColumnsFloatAppearance) property to change the visual appearance of the frame to be shown on the Columns panel.
- Background(exColumnsFloatBackColor) property specifies the color to be shown on the Columns panel background.
- Background(exColumnsFloatCaptionBackColor/exColumnsFloatCaptionForeColor) property specifies the visual appearance of the Columns' caption and the foreground color.
- Background(exColumnsFloatCloseButton) property specifies the visual appearance of the Close button to be shown on the Columns caption.

- Background(exColumnsFloatScrollBackColor) property specifies the visual appearance of the scroll bars to be visible in the Columns panel.
- Background(exColumnsFloatScrollPressBackColor) property specifies the visual appearance of the scroll bars to be visible in the Columns panel, when the scroll button is pressed.
- Background(exColumnsFloatScrollDown) property specifies the visual appearance Down scroll bar.
- Background(exColumnsFloatScrollUp) property specifies the visual appearance Up scroll bar.

property Pivot.PivotColumnsSortOrder as PivotColumnsSortOrderEnum

Specifies the sorting order for the columns being shown in the control's columns floating panel.

Type	Description
PivotColumnsSortOrderEnum	A PivotColumnsSortOrderEnum expression that defines the sorting order of the columns to be displayed on the Columns Floating Panel.

By default, the PivotColumnsSortOrder property is exPivotColumnsUnsorted. The PivotColumnsSortOrder property specifies the sorting order of the columns to be shown on the Columns Floating Panel. The PivotColumnsFloatBarVisible property shows or hides the control's Columns Floating Panel/Bar. The exPivotBarContextSortAscending, exPivotBarContextSortReverse flags of PivotBarVisible property specifies the sorting order of the columns to be shown on the pivot bar's context menu.

The following screen shot shows the control's Columns Floating Panel:

property Pivot.PivotRows as String

Specifies the list of group-by columns that determines the rows in the list.

Type	Description
String	A String expression that indicates the list of columns that builds the first column / group-by column in the control's list. The column is identified by its index, and it can be followed by A or D character, which indicates Ascending or Descending sorting order. The , (comma character) may divide multiple columns in the list. For instance, the "*0,1:D[bold]*" indicates that the Group-By columns are 0 and 1, and the 1 is in descending order displayed in bold. If A is missing, the default sorting order is ascending.

The PivotRows property specifies the list of columns that builds the first column in the control's list. The LockRowsColumn property specifies whether the first column is locked or unlocked. By lock a row or a column it means that the position of it remains unchanged when the user scrolls the control's content. Use the Layout property to save or restore the control's view once the user closes/runs the application. The DataColumns collection holds the collection of Column objects that specifies the control's data. The FormatConditionalAppearances helps you to provide conditional-format for your data, or in other words, ability to highlight values that matches a specified expression.

The PivotRows in **BNF** notation is:

```
PivotRows ::= "<Column>[,<Column>]"
Column ::= <Index>[:<Order>][<Options>]
Index ::= <Digit>[<Digit>]
Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Order ::= A | D
Options ::= \[<Option> | ,<Option>\]
Option ::= <CheckOption> | content=<RadioOption>
```

where

- Index, is the index of the Column, a value between 0 and DataColumns.Count - 1
- CheckOption is any Key in the FormatAppearances collection
- RadioOption is any Key in the FormatContents collection

The control can load data using one of the following methods:

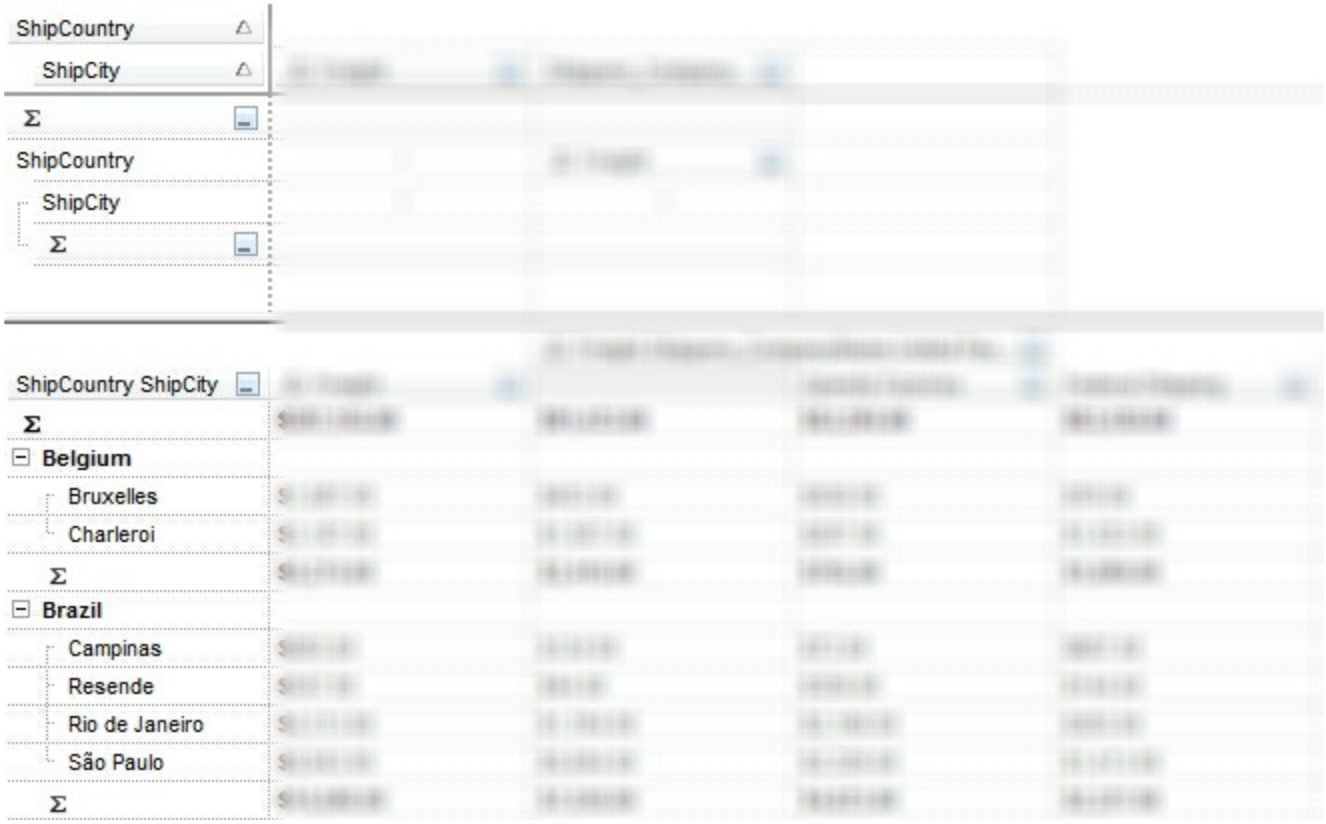
- Import method, loads data from a CSV file or from specified text.

- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method
- The user can drag and drop any TXT or XML files to the control.

Once the data is loaded to the control, the user can drag and drop columns to summarize the data or set the following properties in the following order:

- PivotRows property, specifies the list of columns that determines the first column in the control's list
- PivotColumns property, specifies the list of columns that determines the rest of the columns in the control's list
- PivotTotals property, specifies the list of aggregate functions to be displayed in the control's list

The following screen shot shows the first column in the control's list, that's specified by the PivotRows property:



property Pivot.PivotTotalDefaultFormatAppearances as String

Specifies the list of format-appearances (key of FormatAppearance object), separated by comma, to be applied on the Total field when it is displayed in the pivot-table.

Type	Description
String	A String expression that specifies the list of format-appearances (key of FormatAppearance object), separated by comma, to be applied on the Total field when it is displayed in the pivot-table.

By default, the PivotTotalDefaultFormatAppearances property is empty. The PivotTotalDefaultFormatAppearances property specifies the list of format-appearances (Key of FormatAppearance object, in the FormatAppearances collection), separated by comma, to be applied on the Total field when it is displayed in the pivot-table. The PivotTotalDefaultFormatContent property specifies the default format (key of FormatContent object) to be applied on the Total field when it is displayed in the pivot-table.

property Pivot.PivotTotalDefaultFormatContent as String

Specifies the default format (key of FormatContent object) to be applied on the Total field when it is displayed in the pivot-table.

Type	Description
String	A String expression that specifies the default format (key of FormatContent object) to be applied on the Total field when it is displayed in the pivot-table.

By default, the PivotTotalDefaultFormatContent property is empty. The PivotTotalDefaultFormatContent property Specifies the default format (Key of FormatContent object, in the FormatContents collection) to be applied on the Total field when it is displayed in the pivot-table. The PivotTotalDefaultFormatAppearances property specifies the list of format-appearances (Key of FormatAppearance object, in the FormatAppearances collection), separated by comma, to be applied on the Total field when it is displayed in the pivot-table.

property Pivot.PivotTotals as String

Indicates the list of totals/subtotals to be shown in the list.

Type	Description
String	A String expression that shows the total/subtotals field to be displayed. For instance the "*sum(0)[bold]/sum[bold]*" displays the grand total in bold on the top of the control, and the SUM for all values of the Column with the index 0

The PivotTotals property specifies the total/aggregate rows. The Aggregates collection holds a collection of Aggregate functions the user can display to summarize the data. The SelectableAggregateRows property specifies whether the inside total/aggregate rows are selectable or unselectable. Use the FormatPivotAggregate / FormatPivotTotal property to display aggregate functions in a different format.

The PivotTotals in **BNF** notation is:

```
PivotTotals ::= "<Part> | <Part>/<Part>"
Part ::= "<Aggregate>[,<Aggregate>]"
Aggregate ::= <AggregateKey>[<Options>] | <AggregateKey>(<Index>)
[<Options>]
Index ::= <Digit>[<Digit>]
Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Options ::= \[<Option>|,<Option>\]
Option ::= <CheckOption>|content=<RadioOption>
```

where

- AggregateKey, is any Key in the Aggregates collection
- Index, is the index of the Column, a value between 0 and DataColumns.Count - 1
- CheckOption is any Key in the FormatAppearances collection
- RadioOption is any Key in the FormatContents collection

The control can load data using one of the following methods:

- Import method, loads data from a CSV file or from specified text.
- DataSource property, assigns an ADO/DAO record set to be loaded.
- LoadXML method, loads an XML file previously saved using the SaveXML method
- The user can drag and drop any TXT or XML files to the control.

Once the data is loaded to the control, the user can drag and drop columns to summarize the data or set the following properties in the following order:

- PivotRows property, specifies the list of columns that determines the first column in the control's list
- PivotColumns property, specifies the list of columns that determines the rest of the columns in the control's list
- PivotTotals property, specifies the list of aggregate functions to be displayed in the control's list

property Pivot.RadioImage(Checked as Boolean) as Long

Retrieves or sets a value that indicates the image used by cells of radio type.

Type	Description
Checked as Boolean	A boolean expression that indicates the radio's state. True means checked, and False means unchecked.
Long	A long expression that indicates the index of image used to paint the radio button. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part.

Use RadioImage and CheckImage properties to define the icons used for radio and check box cells. The RadioImage property defines the index of the icon being used by radio buttons. Use the Images method to insert icons at runtime. The ImageSize property defines the size (width/height) of the radio-buttons.

method Pivot.Refresh ()

Refreses the control.

Type	Description

method Pivot.ReplaceIcon ([Icon as Variant], [Index as Variant])

Adds a new icon, replaces an icon or clears the control's image list.

Type	Description
Icon as Variant	A long expression that indicates the icon's handle.
Index as Variant	A long expression that indicates the index where icon is inserted.
Return	**Description**
Long	A long expression that indicates the index of the icon in the images collection

Use the ReplaceIcon property to add, remove or replace an icon in the control's images collection. Also, the ReplaceIcon property can clear the images collection. Use the Images method to attach a image list to the control.

The following VB sample adds a new icon to control's images list:

i = ExPivot1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added

The following VB sample replaces an icon into control's images list::

i = ExPivot1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.

The following VB sample removes an icon from control's images list:

ExPivot1.ReplaceIcon 0, i, i specifies the index of icon removed.

The following VB clears the control's icons collection:

ExPivot1.ReplaceIcon 0, -1

method Pivot.Reset ([IncludeMask as Variant], [ExcludeMask as Variant])

Resets the control's layouts, so no filtering, sorting, ... is applied to the view.

Type	Description
IncludeMask as Variant	A String expression that specifies the mask of properties to be reset. The mask can include the following special characters: • *, indicates any combination of characters • ?, specifies any single character • #, any digit character • space, specifies the separators of multiple patterns.
ExcludeMask as Variant	A String expression that specifies the mask of properties to be keep. The mask can include the following special characters: • *, indicates any combination of characters • ?, specifies any single character • #, any digit character • space, specifies the separators of multiple patterns.

The Reset method resets the control's layout. The Layout property of the control includes information about sorting, filtering, sizing or positions of the columns (data columns). The pivot columns are displayed in the control's pivot bar, while the data columns are displayed on the control's data view. The LayoutStartChanging / LayoutEndChanging events notify your application when user performs operations in the control's pivot bar, for instance, sorts a pivot column.

For instance, the following VB sample resets the position properties when user changes the filter of a pivot column:

```
Private Sub Pivot1_LayoutEndChanging(ByVal Operation As
EXPIVOTLibCtl.LayoutChangingEnum)
   If Operation = exPivotDataColumnFilterChange Then
      Pivot1.Reset "c*.position*"
   End If
End Sub
```

When user changes the pivot columns, like adding new pivot columns, the control keeps the layout of the data view, so the position, size of data columns is keep. The same thing is happen for sorting, filtering and so on.

method Pivot.SaveXML (Destination as Variant)

Saves the control's content as XML document to the specified location, using the MSXML parser.

Type	Description

Destination as Variant	This object can represent a file name, an XML document object, or a custom object that supports persistence as follows:
	
	• String - Specifies the file name. Note that this must be a file name, rather than a URL. The file is created if necessary and the contents are entirely replaced with the contents of the saved document. For example:
	
	`Pivot1.SaveXML("sample.xml")`
	
	• XML Document Object. For example:
	
	`Dim xmldoc as Object`
	`Set xmldoc = CreateObject("MSXML.DOMDocument")`
	`Pivot1.SaveXML(xmldoc)`
	
	• Custom object supporting persistence - Any other custom COM object that supports **QueryInterface** for **IStream**, **IPersistStream**, or **IPersistStreamInit** can also be provided here and the document will be saved accordingly. In the **IStream** case, the **IStream::Write** method will be called as it saves the document; in the **IPersistStream** case, **IPersistStream::Load** will be called with an **IStream** that supports the **Read**, **Seek**, and **Stat** methods.

Return	Description
Boolean	A Boolen expression that specifies whether saving the XML document was ok.

The SaveXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to save the control's data in XML documents. The LoadXML method loads XML documents being created with SaveXML method. The SaveXML method saves each column in **<column>** elements under the **<columns>** collection. The **<items>** xml element saves a collection of **<item>** objects. Each <item> object holds information about an item in the control, including its cells or child items. Each item saves a collection of **<cell>** objects that defines the cell for each column.

The control saves the control's data in XML format like follows:

```
- <Content Author Component Version ...>
  - <Columns>
      <Column Caption Position Width ... />
      <Column Caption Position Width ... />
      ...
    </Columns>
  - <Items>
    - <Item Expanded ...>
        <Cell Value ValueFormat Images Image ... />
        <Cell Value ValueFormat Images Image ... />
        ...
      - <Items>
        - <Item Expanded ...>
        - <Item Expanded ...>
        ....
        </Items>
      </Item>
    </Items>
  </Content>
```

property Pivot.SelBackColor as Color

Retrieves or sets a value that indicates the selection background color.

Type	Description
Color	A color expression that indicates the selection background color. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the SelBackColor property applies the background color only to list area. Use the SelBackColor and SelForeColor properties to define the colors used for selected items. The control highlights the selected items only if the SelBackColor and BackColor properties have different values, and the SelForeColor and ForeColor properties have different values.

For instance, the following VB sample changes the visual appearance for the selected item. The SelBackColor property indicates the selection background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the SelBackColor property to indicates the index of the skin that we want to use. The sample applies the "" to the selected item(s):

```
With Pivot1
   With .VisualAppearance
      .Add &H23, App.Path + "\selected.ebn"
   End With
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = &H23000000
End With
```

The sample adds the skin with the index 35 (Hexa 23), and applies to the selected item using the SelBackColor property.

The following C++ sample applies a new appearance to the selected item(s):

```
#include "Appearance.h"
m_pivot.GetVisualAppearance().Add( 0x23,
COleVariant(_T("D:\\Temp\\ExPivot_Help\\selected.ebn")) );
m_pivot.SetSelBackColor( 0x23000000 );
```

```
m_pivot.SetSelForeColor( 0 );
```

The following VB.NET sample applies a new appearance to the selected item(s):

```
With AxPivot1
   With .VisualAppearance
      .Add(&H23, "D:\Temp\ExPivot_Help\selected.ebn")
   End With
   .SelForeColor = Color.Black
   .Template = "SelBackColor = 587202560"
End With
```

The VB.NET sample uses the Template property to assign a new value to the SelBackColor property. The 587202560 value represents &23000000 in hexadecimal.

The following C# sample applies a new appearance to the selected item(s):

```
axPivot1.VisualAppearance.Add(0x23, "D:\\Temp\\ExPivot_Help\\selected.ebn");
axPivot1.Template = "SelBackColor = 587202560";
```

The following VFP sample applies a new appearance to the selected item(s):

```
With thisform.Pivot1
   With .VisualAppearance
      .Add(35, "D:\Temp\ExPivot_Help\selected.ebn")
   EndWith
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = .587202560
EndWith
```

The 587202560 value represents &23000000 in hexadecimal. The 32 value represents &23 in hexadecimal

How do I assign a new look for the selected item?

The component supports skinning parts of the control, including the selected item. Shortly, the idea is that identifier of the skin being added to the Appearance collection is stored in the first significant byte of property of the color type. In our case, we know that the SelBackColor property changes the background color for the selected item. This is what we need to change. In other words, we need to change the visual appearance for the selected item, and that means changing the background color of the selected item. So, the following

code (blue code) changes the appearance for the selected item:

```
With Pivot1
   .VisualAppearance.Add &H34, App.Path + "\aqua.ebn"
   .SelBackColor = &H34000000
End With
```

Please notice that the 34 hexa value is arbitrary chosen, it is not a predefined value. Shortly, we have added a skin with the identifier 34, and we specified that the SelBackColor property should use that skin, in order to change the visual appearance for the selected item. Also, please notice that the 34 value is stored in the first significant byte, not in other position. For instance, the following sample doesn't use any skin when displaying the selected item:

```
With Pivot1
   .VisualAppearance.Add &H34, App.Path + "\aqua.ebn"
   .SelBackColor = &H34
End With
```

This code (red code) DOESN'T use any skin, because the 34 value is not stored in the higher byte of the color value. The sample just changes the background color for the selected item to some black color (RGB(0,0,34)). So, please pay attention when you want to use a skin and when to use a color. Simple, if you are calling &H**34**000000, you have 34 followed by 6 (six) zeros, and that means the first significant byte of the color expression. Now, back to the problem. The next step is how we are creating skins? or EBN files? The Exontrol's exbutton component includes a builder tool that saves skins to EBN files. So, if you want to create new skin files, you need to download and install the exbutton component from our web site. Once that the exbutton component is installed, please follow the steps.

Let's say that we have a BMP file, that we want to stretch on the selected item's background.

1. Open the VB\Builder or VC\Builder sample
2. Click the **New File** button (on the left side in the toolbar), an empty skin is created.
3. Locate the **Background** tool window and select the **Picture\Add New** item in the menu, the Open file dialog is opened.
4. Select the picture file (GIF, BMP, JPG, JPEG). You will notice that the visual appearance of the focused object in the skin is changed, actually the picture you have selected is tiled on the object's background.
5. Select the **None** item, in the Background tool window, so the focused object in the skin is not displaying anymore the picture being added.

6. Select the **Root** item in the skin builder window (in the left side you can find the hierarchy of the objects that composes the skin), so the Root item is selected, and so focused.
7. Select the picture file you have added at the step 4, so the Root object is filled with the picture you have chosen.
8. Resize the picture in the Background tool window, until you reach the view you want to have, no black area, or change the CX and CY fields in the Background tool window, so no black area is displayed.
9. Select **Stretch** button in the Background tool window, so the Root object stretches the picture you have selected.
10. Click the **Save a file** button, and select a name for the new skin, click the Save button after you typed the name of the skin file. Add the .ebn extension.
11. Close the builder

You can always open the skin with the builder and change it later, in case you want to change it.

Now, create a new project, and insert the component where you want to use the skin, and add the skin file to the Appearance collection of the object, using blue code, by changing the name of the file or the path where you have selected the skin. Once that you have added the skin file to the Appearance collection, you can change the visual appearance for parts of the controls that supports skinning. **Usually the properties that changes the background color for a part of the control supports skinning as well.**

property Pivot.SelBackMode as BackModeEnum

Retrieves or sets a value that indicates whether the selection is transparent or opaque.

Type	Description
BackModeEnum	A BackModeEnum expression that indicates whether the selection is transparent or opaque.

Use the SelBackMode property to specify how the selection appears. Use the SelBackMode property to specify how the control displays the selection when the control has a picture on its background. Use the SelBackColor property to specify the selection background color. Use the SelForeColor property to specify the selection foreground color.

property Pivot.SelectableAggregateRows as Boolean

Specifies whether the aggregate rows are selectable or un-selectable.

Type	Description
Boolean	A Boolean expression that specifies whether the total/aggregate rows are selectable or non-selectable/

By default, the SelectableAggregateRows property is True, which means that the user can select the total/aggregate rows. The PivotTotals property specifies the total/aggregate rows. The LockTotalRows property specifies whether the grand total rows are locked or unlocked. The LockRowsColumn property specifies whether the first column is locked or unlocked. By lock a row or a column it means that the position of it remains unchanged when the user scrolls the control's content.

method Pivot.SelectAll ()

Selects all rows.

Type	Description

property Pivot.SelectOnRelease as Boolean

Indicates whether the selection occurs when the user releases the mouse button.

Type	Description
Boolean	A Boolean expression that indicates whether the selection occurs when the user releases the mouse button.

By default, the SelectOnRelease property is False. By default, the selection occurs, as soon as the user clicks an object. The SelectOnRelease property indicates whether the selection occurs when the user releases the mouse button.

property Pivot.SelForeColor as Color

Retrieves or sets a value that indicates the selection foreground color.

Type	Description
Color	A color expression that indicates the selection foreground color.

By default, the SelForeColor property is applied ONLY to selected items being displayed in the list area. Use the SelForeColor and SelBackColor properties to change the colors used for selected items. The control highlights the selected items only if the SelBackColor and BackColor properties have different values, and the SelForeColor and ForeColor properties have different values.

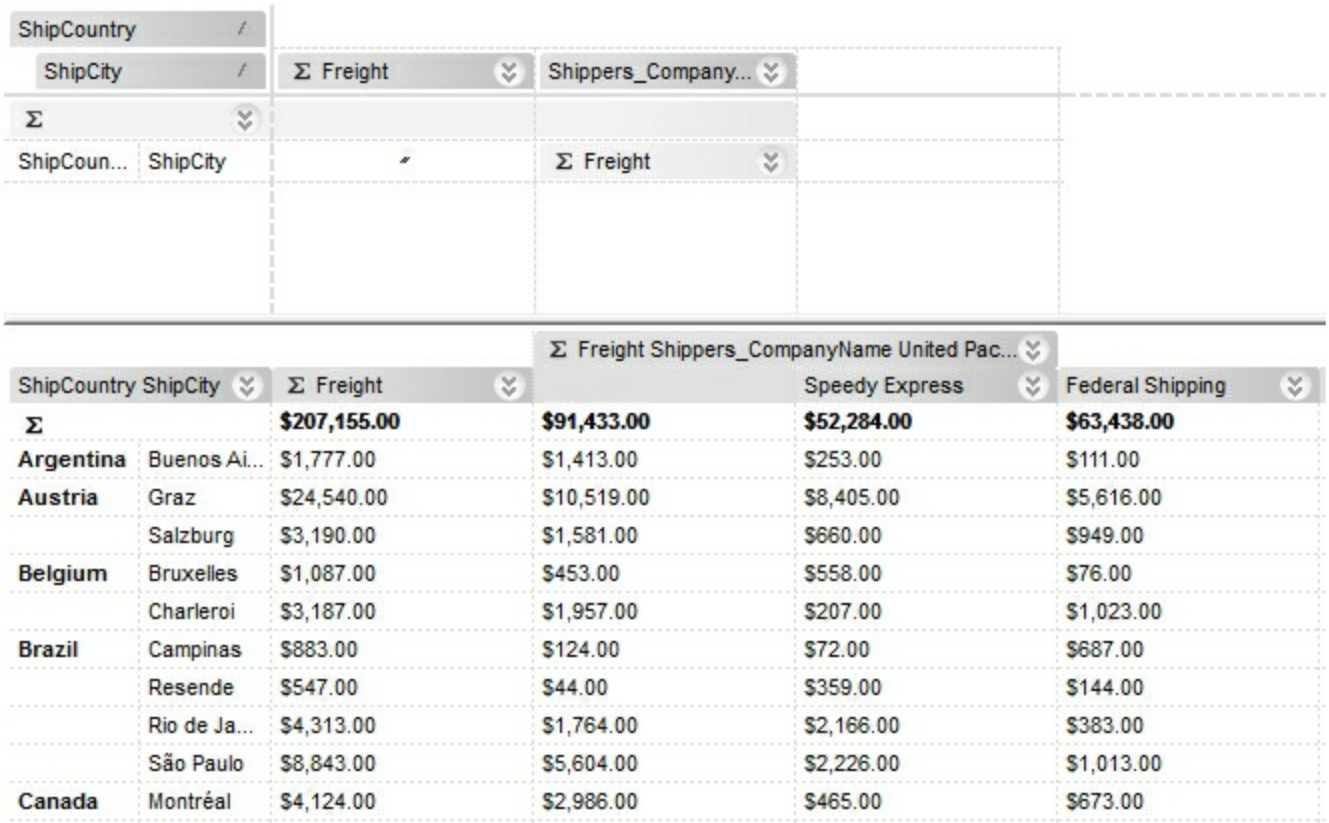
property Pivot.ShowBranchRows as ShowBranchRowsEnum

Indicates how the branch rows displays the information (divider items).

Type	Description
ShowBranchRowsEnum	A ShowBranchRowsEnum expression that specifies the way the rows are being shown in the control's list.

By default, the ShowBranchRows property is exBranchTree. Use the ShowBranchRows property to let the control displays the rows in a compact way as follows. The HasLines property enhances the graphic representation of a grid control's hierarchy by drawing lines that link child items to their corresponding parent item. The LinesAtRoot property link items at the root of the hierarchy. The DrawGridLines property specifies whether the control displays the grid lines. Use the ShowViewCompact property on exViewCompact to summarize multiple aggregate functions on the same cell.

The following screen shot shows the data using the ShowBranchRows property on **exBranchCompact**:



The following screen shot shows the data using the ShowBranchRows property on **exBranchTree** (default):

ShipCountry /			
ShipCity /	Σ Freight	Shippers_Company...	
Σ			
ShipCountry		Σ Freight	
ShipCity			
Σ			

		Σ Freight Shippers_CompanyName United Pac...		
ShipCountry ShipCity	Σ Freight		Speedy Express	Federal Shipping
Σ	**$207,155.00**	**$91,433.00**	**$52,284.00**	**$63,438.00**
Argentina				
Buenos Aires	$1,777.00	$1,413.00	$253.00	$111.00
Σ	**$1,777.00**	**$1,413.00**	**$253.00**	**$111.00**
Austria				
Graz	$24,540.00	$10,519.00	$8,405.00	$5,616.00
Salzburg	$3,190.00	$1,581.00	$660.00	$949.00
Σ	**$27,730.00**	**$12,100.00**	**$9,065.00**	**$6,565.00**
Belgium				
Bruxelles	$1,087.00	$453.00	$558.00	$76.00

The following screen shot shows the data using the ShowBranchRows property on **exBranchTree + exBranchRowDivider**:

ShipCountry /			
ShipCity /	Σ Freight	Shippers_Company...	
Σ			
ShipCountry		Σ Freight	
ShipCity			
Σ			

		Σ Freight Shippers_CompanyName United Pac...		
ShipCountry ShipCity	Σ Freight		Speedy Express	Federal Shipping
Σ	**$207,155.00**	**$91,433.00**	**$52,284.00**	**$63,438.00**
Argentina				
Buenos Aires	$1,777.00	$1,413.00	$253.00	$111.00
Σ	**$1,777.00**	**$1,413.00**	**$253.00**	**$111.00**
Austria				
Graz	$24,540.00	$10,519.00	$8,405.00	$5,616.00
Salzburg	$3,190.00	$1,581.00	$660.00	$949.00
Σ	**$27,730.00**	**$12,100.00**	**$9,065.00**	**$6,565.00**
Belgium				
Bruxelles	$1,087.00	$453.00	$558.00	$76.00
Charleroi	$3,187.00	$1,957.00	$207.00	$1,023.00

The following screen shot shows the data using the ShowBranchRows property on **exBranchTree + exBranchIncludeAggregate**:

		Σ Freight		
ShipCountry ShipCity	Σ Freight	United Packa...	Speedy Expr...	Federal Shipping
Argentina	**$1,890.00**	**$438.00**	**$1,218.00**	**$234.00**
Buenos Aires	$1,890.00	$438.00	$1,218.00	$234.00
Austria	**$3,730.00**	**$3,086.00**	**$1,914.00**	**($1,270.00)**
Graz	$2,789.00	$2,452.00	$1,680.00	($1,343.00)
Salzburg	$941.00	$634.00	$234.00	$73.00
Belgium	**$3,098.00**	**$1,508.00**	**$516.00**	**$1,074.00**
Bruxelles	$1,085.00	$608.00	$219.00	$258.00
Charleroi	$2,013.00	$900.00	$297.00	$816.00
Brazil	**$3,316.00**	**$1,112.00**	**$382.00**	**$1,822.00**
Campinas	($1,098.00)	($2,638.00)	$849.00	$691.00
Resende	$1,441.00	$1,139.00	$174.00	$128.00
Rio de Janeiro	$2,931.00	$1,159.00	$1,251.00	$521.00
São Paulo	$42.00	$1,452.00	($1,892.00)	$482.00

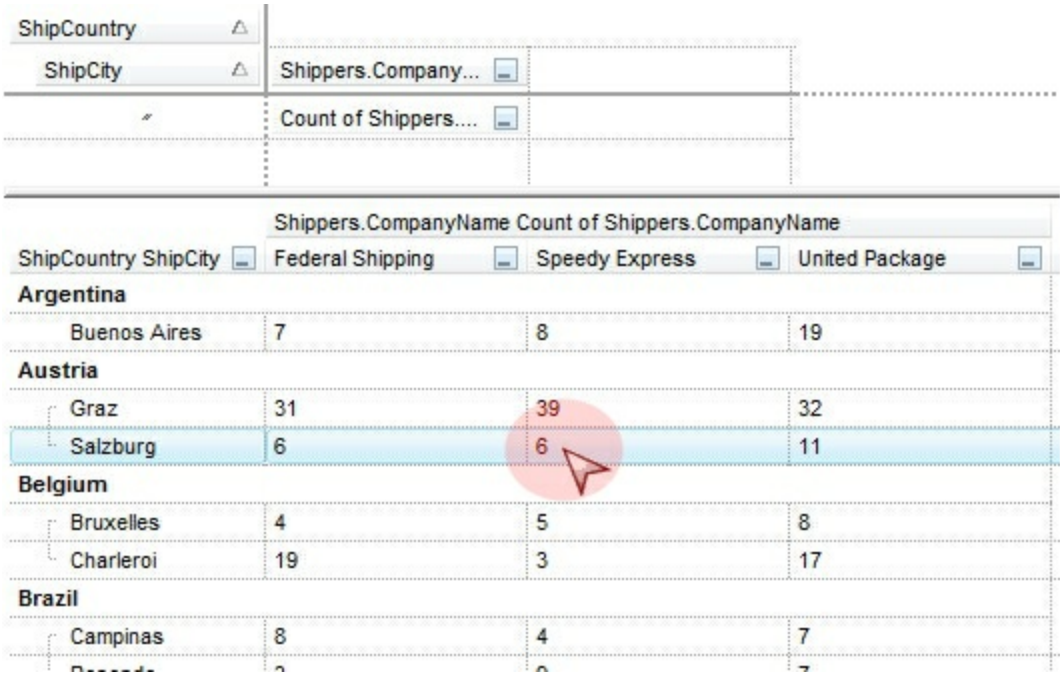
property Pivot.ShowDataOnDblClick as Boolean

Specifies whether the user shows the original data that generated the result when user double clicks a cell.

Type	Description
Boolean	A boolean expression that specifies whether the data that generated the result is shown once the user double click the cell.

By default, the ShowDataOnDblClick property is False. Use the ShowDataOnDblClick property on True, to let the user to display the data that generated the result when a cell is double clicked. For instance, you double click a cell and the list will be filled with the data that generated the result being double clicked. Use the ExpandOnDblClick property to expand an item once the user double clicks the item. The same property allows you to display more data if the user double clicks a ... row, while the DisplayPivotData property is positive.

The following screen shot shows the data before double clicking the red cell (Speedy Express/Salzburg/Austria):



The following screen shot shows the data that generated the result (6) after double clicking the red cell (Speedy Express/Salzburg/Austria):

ShipCountry		
ShipCity	Shippers.Company...	
	Count of Shippers....	

T...

ShipCity	ShipCountry	Shippers.CompanyName	Order Details.OrderID	ProductID	ProductName	UnitPrice
Salzburg	Austria	Speedy Express	10686	17	Alice Mutton	39
Salzburg	Austria	Speedy Express	10686	26	Gumbär Gummibärchen	31.23
Salzburg	Austria	Speedy Express	10747	31	Gorgonzola Telino	12.5
Salzburg	Austria	Speedy Express	10747	69	Gudbrandsdalsost	36
Salzburg	Austria	Speedy Express	10747	63	Vegie-spread	43.9
Salzburg	Austria	Speedy Express	10747	41	Jack's New England Cl...	9.65

As you can see, the list displays only the data of Speedy Express/Salzburg/Austria, so a 6 rows.

property Pivot.ShowImageList as Boolean

Specifies whether the control's image list window is visible or hidden.

Type	Description
Boolean	A boolean expression that specifies whether the control's image list window is visible or hidden.

By default, the ShowImageList property is True. Use the ShowImageList property to hide the control's images list window. The control's images list window is visible only at design time. Use the Images method to associate an images list control to the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. Use the CheckImage or RadioImage property to specify a different look for checkboxes or radio buttons in the cells.

method Pivot.ShowToolTip (ToolTip as String, [Title as Variant], [Alignment as Variant], [X as Variant], [Y as Variant])

Shows the specified tooltip at given position.

Type	Description
ToolTip as String	The ToolTip parameter can be any of the following: • NULL(BSTR) or "<null>"(string) to indicate that the tooltip for the object being hovered is not changed • A String expression that indicates the description of the tooltip, that supports built-in HTML format (adds, replaces or changes the object's tooltip)
Title as Variant	The Title parameter can be any of the following: • missing (VT_EMPTY, VT_ERROR type) or "<null>" (string) the title for the object being hovered is not changed. • A String expression that indicates the title of the tooltip (no built-in HTML format) (adds, replaces or changes the object's title)
Alignment as Variant	A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing (VT_EMPTY, VT_ERROR) the alignment of the tooltip for the object being hovered is not changed. The Alignment parameter can be one of the following: • 0 - exTopLeft • 1 - exTopRight • 2 - exBottomLeft • 3 - exBottomRight • 0x10 - exCenter • 0x11 - exCenterLeft • 0x12 - exCenterRight • 0x13 - exCenterTop • 0x14 - exCenterBottom By default, the tooltip is aligned relative to the top-left corner (0 - exTopLeft).

X as Variant	Specifies the horizontal position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current horizontal position of the cursor (current x-position) • a numeric expression that indicates the horizontal screen position to show the tooltip (fixed screen x-position) • a string expression that indicates the horizontal displacement relative to default position to show the tooltip (moved)
Y as Variant	Specifies the vertical position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current vertical position of the cursor (current y-position) • a numeric expression that indicates the vertical screen position to show the tooltip (fixed screen y-position) • a string expression that indicates the vertical displacement relative to default position to show the tooltip (displacement)

Use the ShowToolTip method to display a custom tooltip at specified position or to update the object's tooltip, title or position. You can call the ShowToolTip method during the MouseMove event. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

For instance:

- ShowToolTip(`<null>`,`<null>`,,`+8`,`+8`), shows the tooltip of the object moved relative

to its default position

- ShowToolTip(`<null>`,`new title`), adds, changes or replaces the title of the object's tooltip
- ShowToolTip(`new content`), adds, changes or replaces the object's tooltip
- ShowToolTip(`new content`,`new title`), shows the tooltip and title at current position
- ShowToolTip(`new content`,`new title`,,`+8`,`+8`), shows the tooltip and title moved relative to the current position
- ShowToolTip(`new content`,``,,128,128), displays the tooltip at a fixed position
- ShowToolTip(``,``), hides the tooltip

The ToolTip parameter supports the built-in HTML format like follows:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).

- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the

height of the font. For instance the "<**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:

property Pivot.ShowViewCompact as ShowViewCompactEnum

Indicates whether the view compacts the data being displayed.

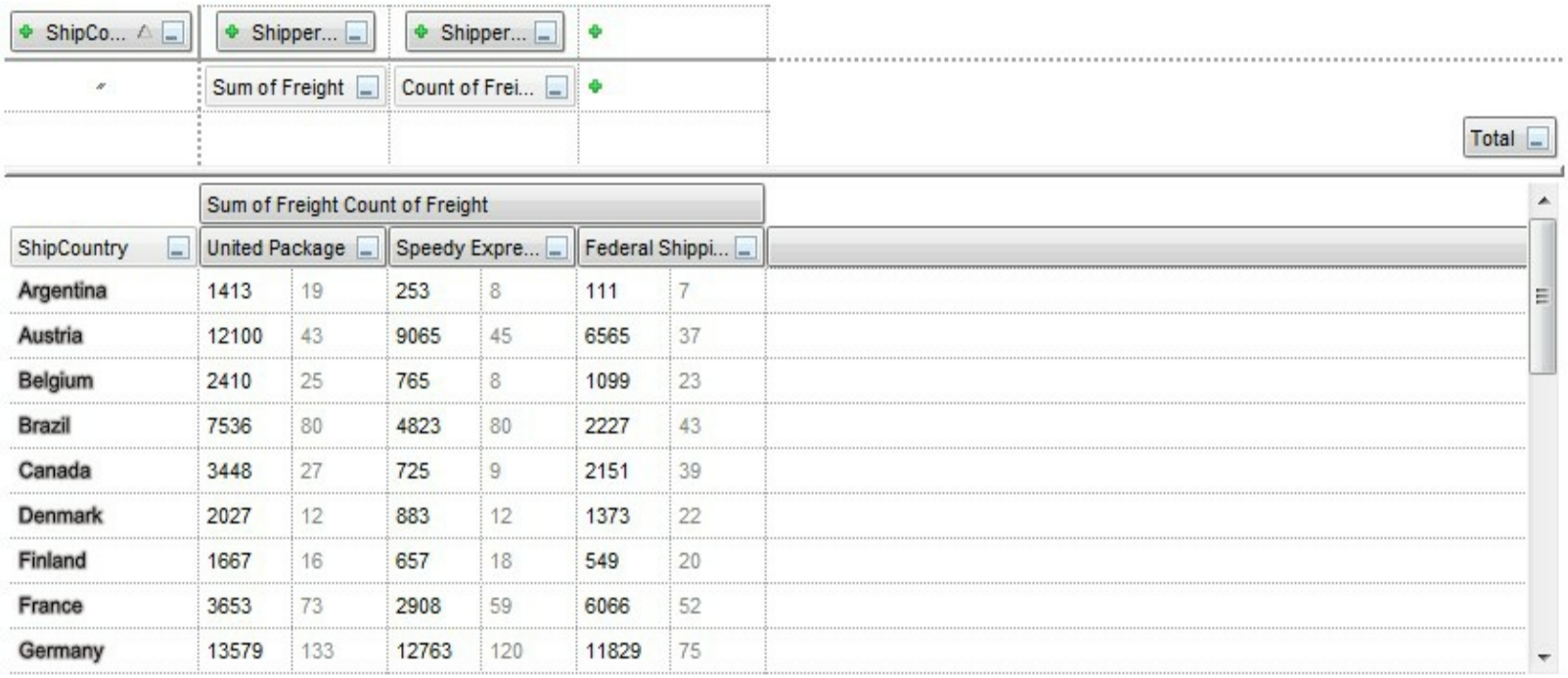
Type	Description
ShowViewCompactEnum	A ShowViewCompactEnum expression that specifies the way the view compacts the displaying data.

By default, the ShowViewCompact property is exViewNotCompact, which means that the property has no effect. Use the ShowViewCompact property on exViewCompact to summarize multiple aggregate functions on the same cell. For instance, if you need to display the price and the quantity sold in the same cell. At runtime, if the ShowViewCompact property is not exViewNotCompact, the control may include less columns than when the ShowViewCompact property is exViewNotCompact. The ShowBranchRows property indicates how the branch rows displays the information (divider items). The DrawGridLines property specifies whether the control displays the grid lines.

The following movies show how Compacting works:

- The movie shows how you can add several aggregate functions to the same cell, or how you can compact the view to display several aggregate functions to the same cell.

The following screen shot displays data compacted, ShowViewCompact property is exViewCompact:



The following screen shot displays the same data un-compacted, ShowViewCompact property is exViewNotCompact:

ShipCo... | Shipper... | Shipper...

Sum of Freight | Count of Frei...

Total

	Sum of Freight			Count of Freight		
ShipCountry	United Package	Speedy Expre...	Federal Shippi...	United Package	Speedy Expre...	Federal Shippi...
Argentina	1413	253	111	19	8	7
Austria	12100	9065	6565	43	45	37
Belgium	2410	765	1099	25	8	23
Brazil	7536	4823	2227	80	80	43
Canada	3448	725	2151	27	9	39
Denmark	2027	883	1373	12	12	22
Finland	1667	657	549	16	18	20
France	3653	2908	6066	73	59	52
Germany	13579	12763	11829	133	120	75

property Pivot.SingleSel as Boolean

Retrieves or sets a value that indicates whether the control supports single or multiple selection.

Type	Description
Boolean	A boolean expression that indicates whether the control support single or multiple selection.

The SingleSel property specifies whether the control support single or multiple selection. By default, the SingleSel property is True, and so only a single item can be selected. Use the SelBackColor and SelForeColor properties to specify the background and foreground colors for selected items.

property Pivot.Statistics as String

Gives statistics data of objects being hold by the control.

Type	Description
String	A String expression that indicates a few statistics information about the current data.

The Statistics property gives statistics data of objects being hold by the control.

property Pivot.Template as String

Specifies the control's template.

Type	Description
String	A string expression that defines the control's template

The control's template uses the X-Script language to initialize the control's content. Use the Template property page of the control to update the control's Template property. Use the Template property to execute code by passing instructions as a string (template string). Use the ExecuteTemplate property to get the result of executing a template script.

Most of our UI components provide a Template page that's accessible in design mode. No matter what programming language you are using, you can have a quick view of the component's features using the WYSWYG Template editor.

- Place the control to your form or dialog.
- Locate the Properties item, in the control's context menu, in design mode. If your environment doesn't provide a Properties item in the control's context menu, please try to locate in the Properties browser.
- Click it, and locate the Template page.
- Click the Help button. In the left side, you will see the component, in the right side, you will see a x-script code that calls methods and properties of the control.

The control's Template page helps user to initialize the control's look and feel in design mode, using the x-script language that's easy and powerful. The Template page displays the control on the left side of the page. On the right side of the Template page, a simple editor is displayed where user writes the initialization code. The control's look and feel is automatically updated as soon as the user types new instructions. The Template script is saved to the container persistence (when Apply button is pressed), and it is executed when the control is initialized at runtime. Any component that provides a WYSWYG Template page, provides a Template property. The Template property executes code from a string (template string).

The Template script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters.

An instruction can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values*

separated by commas. (Sample: h = InsertItem(0,"New Child"))

- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)…. *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The Template supports the following general functions:

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier. For instance, the following code creates an ADOR.Recordset and pass it to the control using the DataSource property:*

The following sample loads the Orders table:

```
Dim rs
ColumnAutoResize = False
rs = CreateObject("ADOR.Recordset")
{
Open("Orders","Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\Exontrol\ExPivot\Sample\SAMPLE.MDB", 3, 3 )
}
DataSource = rs
```

property Pivot.TemplateDef as Variant

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters) = Value*, so our controls provide an alternative using the TemplateDef method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

So, calling the TemplateDef property should be as follows:

```
with (Control)
   TemplateDef = [Dim var_Column]
   TemplateDef = var_Column
   Template = [var_Column.Def(4) = 255]
endwith
```

This sample allocates a variable var_Column, assigns the value to the variable (the second call of the TemplateDef), and the Template call uses the var_Column variable (as an object), to call its Def property with the parameter 4.

Let's say we need to define the background color for a specified column, so we need to call the Def(exCellBackColor) property of the column, to define the color for all cells in the column.

The following **VB6** sample shows setting the Def property such as:

```
With Control
   .Columns.Add("Column 1").Def(exCellBackColor) = 255
   .Columns.Add "Column 2"
   .Items.AddItem 0
   .Items.AddItem 1
```

```
    .Items.AddItem 2
End With
```

In **dBASE Plus**, calling the Def(4) has no effect, instead using the TemplateDef helps you to use properly the Def property as follows:

```
local Control,var_Column

Control = form.Activex1.nativeObject
// Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
with (Control)
    TemplateDef = [Dim var_Column]
    TemplateDef = var_Column
    Template = [var_Column.Def(4) = 255]
endwith
Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The equivalent sample for **XBasic in A5**, is as follows:

```
Dim Control as P
Dim var_Column as P

Control = topparent:CONTROL_ACTIVEX1.activex
' Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
Control.TemplateDef = "Dim var_Column"
Control.TemplateDef = var_Column
Control.Template = "var_Column.Def(4) = 255"

Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The samples just call the Column.Def(4) = Value, using the TemplateDef. The first call of TemplateDef property is "Dim var_Column", which indicates that the next call of the TemplateDef will defines the value of the variable var_Column, in other words, it defines the object var_Column. The last call of the Template property uses the var_Column member to use the x-script and so to set the Def property so a new color is being assigned to the column.

The TemplateDef, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)…. *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please

make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

method Pivot.TemplatePut (NewVal as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
NewVal as Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplatePut method / TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters) = Value*, so our controls provide an alternative using the TemplateDef / TemplatePut method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

The TemplateDef, TemplatePut, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments).... *The .(dot) character splits the object from its property. For instance, the*

Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property Pivot.ToolTipDelay as Long

Specifies the time in ms that passes before the ToolTip appears.

Type	Description
Long	A long expression that specifies the time in ms that passes before the ToolTip appears.

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip. Use the ToolTipFont property or <font> HTML element to assign a new font for tooltips.

property Pivot.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

Type	Description
IFontDisp	A Font object being used to display the tooltip.

Use the ToolTipFont property to assign a font for the control's tooltip. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. You can use the <*font*> HTML element, in the tooltip's description to assign a different font for portions of text.

property Pivot.ToolTipPopDelay as Long

Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.

Type	Description
Long	A long expression that specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to assign a font for the control's tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip.

property Pivot.ToolTipWidth as Long

Specifies a value that indicates the width of the tooltip window, in pixels.

Type	Description
Long	A long expression that indicates the width of the tooltip window, in pixels.

Use the ToolTipWidth property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipFont property to assign a font for the control's tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip.

method Pivot.UnselectAll ()

Unselects all rows.

Type	Description

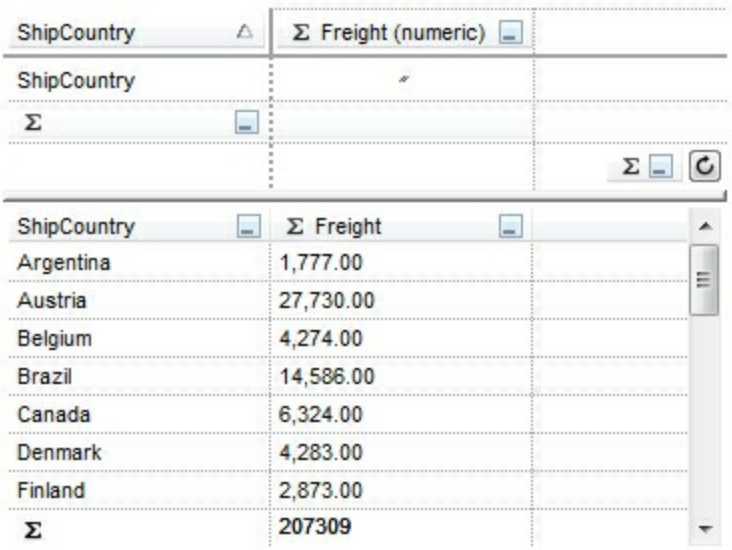
property Pivot.UseVisualTheme as UIVisualThemeEnum

Specifies whether the control uses the current visual theme to display certain UI parts.

Type	Description
UIVisualThemeEnum	An UIVisualThemeEnum expression that specifies which UI parts of the control are shown using the current visual theme.

By default, the UseVisualTheme property is exDefaultVisualTheme, which means that all known UI parts are shown as in the current theme. The UseVisualTheme property may specify the UI parts that you need to enable or disable the current visual theme. The UI Parts are like header, filterbar, check-boxes, buttons and so on. The UseVisualTheme property has effect only a current theme is selected for your desktop. The UseVisualTheme property. Use the Appearance property of the control to provide your own visual appearance using the EBN files.

The following screen shot shows the control while the UseVisualTheme property is exDefaultVisualTheme:



since the second screen shot shows the same data as the UseVisualTheme property is exNoVisualTheme:

ShipCountry	Σ Freight (numeric)
ShipCountry	
Σ	

ShipCountry	Σ Freight
Argentina	1,777.00
Austria	27,730.00
Belgium	4,274.00
Brazil	14,586.00
Canada	6,324.00
Denmark	4,283.00
Finland	2,873.00
Σ	**207309**

property Pivot.ValueFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Variant

Retrieves the value from the point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Variant	A NULL expression (VT_NULL) if no value at the point, or the value from the point.

The ValueFromPoint property gets the value from the cursor, while it hovers the control's list part. The ValueFromPoint(-1,-1) gets the value from the current cursor position, while it hovers the control's list part. The ColumnFromPoint property gets the column from the cursor. The DataColumnFromPoint property retrieves the index of the data column from the point, or -1 if not data column is found.

property Pivot.Version as String

Retrieves the control's version.

Type	Description
String	A string expression that indicates the control's version.

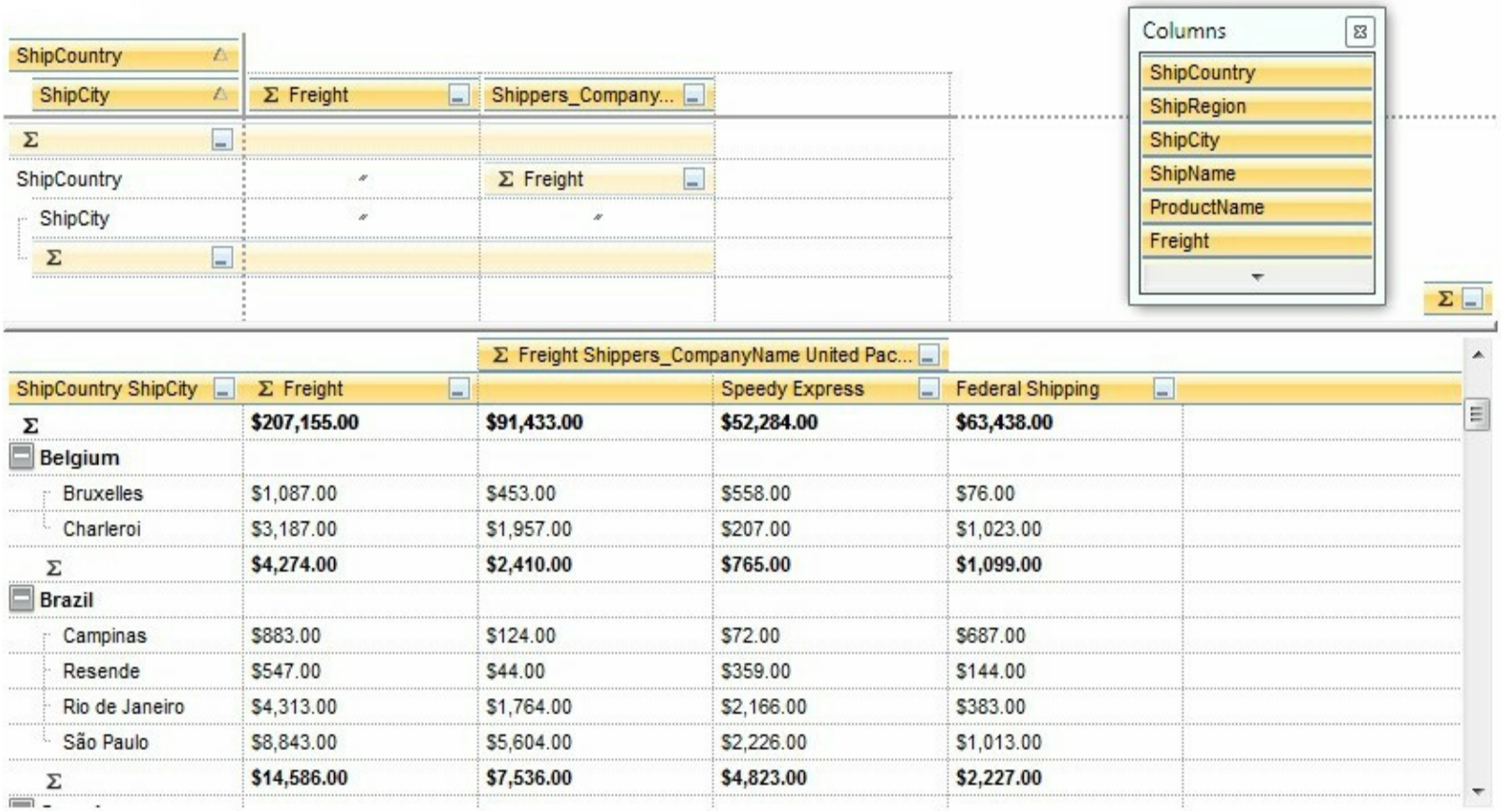
The version property specifies the control's version.

property Pivot.VisualAppearance as Appearance

Retrieves the control's appearance.

Type	Description
Appearance	An Appearance object that holds a collection of skins.

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part.



The skin method may change the visual appearance for the following parts in the control:

- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- **selected item** or cell, SelBackColor property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, and so on, Background property

property Pivot.VisualDesign as String

Invokes the control's VisualAppearance designer.

Type	Description
String	A String expression that encodes the control's Visual Appearance.

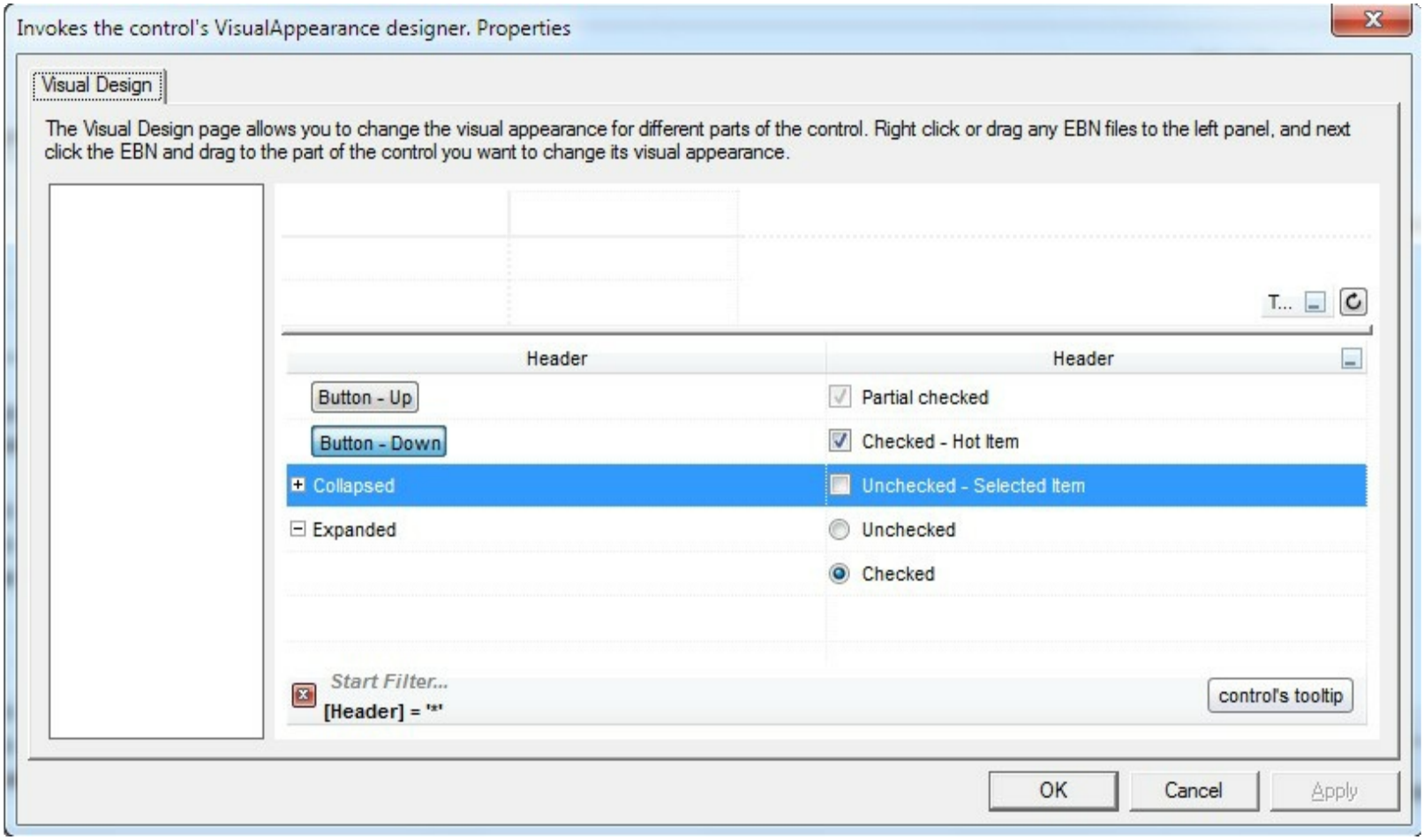
By default, the VisualDesign property is "". The VisualDesign property helps you to define fast and easy the control's visual appearance using the XP-Theme elements or EBN objects. The VisualDesign property can be accessed on design mode, and it can be used to design the visual appearance of different parts of the control by drag and drop XP or EBN elements. The VisualAppearance designer returns an encoded string that can be used to define different looks, just by calling the VisualDesign = encoded_string. If you require removing the current visual appearance, you can call the VisualDesign on "" (empty string). The VisualDesign property encodes EBN or XP-Theme nodes, using the Add method of the Appearance collection being accessed through the VisualAppearance property.

- For the /COM version, click the control in Design mode, select the Properties, and choose the "Visual Design" page.
- For the /NET version, select the VisualDesign property in the Properties browser, and then click ... so the "Visual Design" page is displayed.
- The /WPF version does not provide a VisualAppearance designer, instead you can use the values being generated by the /COM or /NET to apply the same visual appearance.

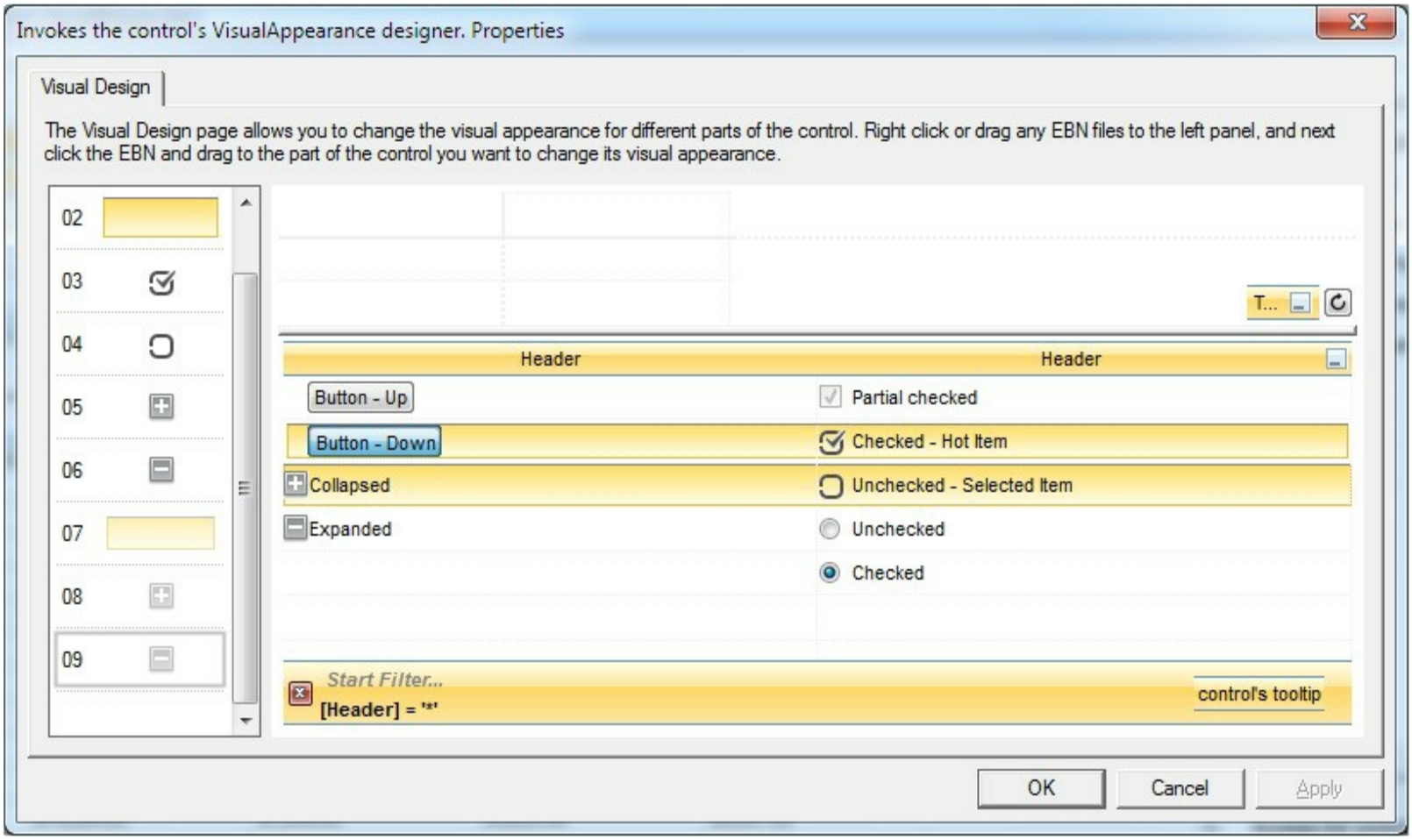
- Click here to watch a movie on how you define the control's visual appearance using the XP-Theme
- Click here to watch a movie on how you define the control's visual appearance using the EBN files.

The left panel, should be user to add your EBN or XP-Theme elements. Once you add them drag and drop the EBN or XP-Theme element from the left side to the part which visual appearance you want to change.

The following picture shows the control's VisualDesign form (empty):



The following picture shows the control's VisualDesign form after applying some EBN objects:



This layout generates the following code:

```
With Pivot1
   .VisualDesign =
"gBFLBWIgBAEHhEJAEGg7oB0HBSQAwABsIfj/jEJAcKhYEjgCAscA8ThQBA8cAgIjgDh8KBAPj
 & _

"RuF6FxmAkchiheZg5gYZIW0yMhZhqD55jIboamcCY2HGG5nCmVh0h2ZYUAyCQ4Xqbh9h8
 & _

"o5B8MwE4HsD4/g/ijHQHoLwrxUjrH0H4Z4rR2h7A8N8UggRNBnGCP8eA/A/gXGSPMfg3w
 & _

"DCDgJQFICxhDQGYBofYQYFCwD4J+XYQwIBECiCwJIExhhnCIDoNAnhzj8CyBcIosQ+BlAwM
 & _

"J8YQlwaBMCaCMd6hRnBpE+HoIwIQ9hdEKM8VYawoCcC8BUSYtxqBuDuFsOwTgLgLhZhAh
 & _

"zhGhtoEB+AsArhnhLhehUB5BfA4BfARBPgWB9h3hhBZB/AvA+BzhkhLhCh7hPg8g1BfhzAKE
 & _

"hQH1hSgAgcAmghgIg2AugLBigiBqAnAzBiVdgIA1ANAjBEgbAmAJMwA+gLgjgyBWA4A0E
 & _

"IAUgCA0AMhjA0ggWUgjh+GhBihI1yAKhiByBqAkV1gCAKAiV3141516g+Jmhj19V+V/AI2A

End With
```

If running the empty control we get the following picture:

ShipCountry
ShipCity | Σ Freight | Shippers_Company...
Σ
ShipCountry | Σ Freight
ShipCity
Σ

Columns
ShipCountry
ShipRegion
ShipCity
ShipName
ProductName
Freight

Σ Freight Shippers_CompanyName United Pac...

ShipCountry ShipCity	Σ Freight		Speedy Express	Federal Shipping
Σ	**$207,155.00**	**$91,433.00**	**$52,284.00**	**$63,438.00**
Belgium				
Bruxelles	$1,087.00	$453.00	$558.00	$76.00
Charleroi	$3,187.00	$1,957.00	$207.00	$1,023.00
Σ	**$4,274.00**	**$2,410.00**	**$765.00**	**$1,099.00**
Brazil				
Campinas	$883.00	$124.00	$72.00	$687.00
Resende	$547.00	$44.00	$359.00	$144.00
Rio de Janeiro	$4,313.00	$1,764.00	$2,166.00	$383.00
São Paulo	$8,843.00	$5,604.00	$2,226.00	$1,013.00
Σ	**$14,586.00**	**$7,536.00**	**$4,823.00**	**$2,227.00**

If running the control using the code being generated by the VisualAppearance designer we get:

ShipCountry
ShipCity | Σ Freight | Shippers_Company...
Σ
ShipCountry | Σ Freight
ShipCity
Σ

Columns
ShipCountry
ShipRegion
ShipCity
ShipName
ProductName
Freight

Σ Freight Shippers_CompanyName United Pac...

ShipCountry ShipCity	Σ Freight		Speedy Express	Federal Shipping
Σ	**$207,155.00**	**$91,433.00**	**$52,284.00**	**$63,438.00**
Belgium				
Bruxelles	$1,087.00	$453.00	$558.00	$76.00
Charleroi	$3,187.00	$1,957.00	$207.00	$1,023.00
Σ	**$4,274.00**	**$2,410.00**	**$765.00**	**$1,099.00**
Brazil				
Campinas	$883.00	$124.00	$72.00	$687.00
Resende	$547.00	$44.00	$359.00	$144.00
Rio de Janeiro	$4,313.00	$1,764.00	$2,166.00	$383.00
São Paulo	$8,843.00	$5,604.00	$2,226.00	$1,013.00
Σ	**$14,586.00**	**$7,536.00**	**$4,823.00**	**$2,227.00**

ExPivot events

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {5C9DF3D3-81B1-42C4-BED6-658F17748686}. The object's program identifier is: "Exontrol.Pivot". The /COM object module is: "ExPivot.dll"

The ExPivot component supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user presses and then releases the left mouse button over the control.
DblClick	Occurs when the user dblclk the left mouse button over an object.
Event	Notifies the application once the control fires an event.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
LayoutEndChanging	Notifies your application once the control's layout has been changed.
LayoutStartChanging	Occurs when the control's layout is about to be changed.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
RClick	Occurs once the user right clicks the control.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Type	Description
AnchorID as String	A string expression that indicates the identifier of the anchor.
Options as String	A string expression that specifies options of the anchor element.

The control fires the AnchorClick event to notify that the user clicks an anchor element. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The AnchorClick event is fired only if prior clicking the control it shows the hand cursor. For instance, if the cell is disabled, the hand cursor is not shown when hovers the anchor element, and so the AnchorClick event is not fired. Use the FormatAnchor property to specify the visual effect for anchor elements. For instance, if the user clicks the anchor *<a1>anchor*, the control fires the AnchorClick event, where the AnchorID parameter is 1, and the Options parameter is empty. Also, if the user clicks the anchor *<a 1;yourextradata>anchor*, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "yourextradata". Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor.

Syntax for AnchorClick event, **/NET** version, on:

C#

```
private void AnchorClick(object sender,string AnchorID,string Options)
{
}
```

VB

```
Private Sub AnchorClick(ByVal sender As System.Object,ByVal AnchorID As
String,ByVal Options As String) Handles AnchorClick
End Sub
```

Syntax for AnchorClick event, **/COM** version, on:

C#

```
private void AnchorClick(object sender,
AxEXPIVOTLib._IPivotEvents_AnchorClickEvent e)
{
}
```

C++

```
void OnAnchorClick(LPCTSTR AnchorID,LPCTSTR Options)
{
}
```

C++ Builder

```
void __fastcall AnchorClick(TObject *Sender,BSTR AnchorID,BSTR Options)
{
}
```

Delphi

```
procedure AnchorClick(ASender: TObject; AnchorID : WideString;Options :
WideString);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AnchorClick(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_AnchorClickEvent);
begin
end;
```

Powe…

```
begin event AnchorClick(string AnchorID,string Options)
end event AnchorClick
```

VB.NET

```
Private Sub AnchorClick(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_AnchorClickEvent) Handles AnchorClick
End Sub
```

VB6

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VBA

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VFP

```
LPARAMETERS AnchorID,Options
```

Xbas…

```
PROCEDURE OnAnchorClick(oPivot,AnchorID,Options)
RETURN
```

Syntax for AnchorClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AnchorClick(AnchorID,Options)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AnchorClick(AnchorID,Options)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAnchorClick String llAnchorID String llOptions
   Forward Send OnComAnchorClick llAnchorID llOptions
End_Procedure
```

Visual Objects

```
METHOD OCX_AnchorClick(AnchorID,Options) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AnchorClick(str _AnchorID,str _Options)
{
}
```

XBasic

```
function AnchorClick as v (AnchorID as C,Options as C)
end function
```

dBASE

```
function nativeObject_AnchorClick(AnchorID,Options)
return
```

event Click ()

Occurs when the user presses and then releases the left mouse button over the Pivot control.

Type	Description

The Click event is fired when the user releases the left mouse button over the control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. The AnchorClick event notifies your application that the user clicks an <a> anchor element.

Syntax for Click event, **/NET** version, on:

C#

```
private void Click(object sender)
{
}
```

VB

```
Private Sub Click(ByVal sender As System.Object) Handles Click
End Sub
```

Syntax for Click event, **/COM** version, on:

C#

```
private void ClickEvent(object sender, EventArgs e)
{
}
```

C++

```
void OnClick()
{
}
```

C++ Builder

```
void __fastcall Click(TObject *Sender)
{
}
```

Delphi

```
procedure Click(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure ClickEvent(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event Click()
end event Click
```

VB.NET

```
Private Sub ClickEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ClickEvent
End Sub
```

VB6

```
Private Sub Click()
End Sub
```

VBA

```
Private Sub Click()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnClick(oPivot)
RETURN
```

Syntax for Click event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Click()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Click()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComClick
    Forward Send OnComClick
End_Procedure
```

Visual Objects

```
METHOD OCX_Click() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Click()
{
}
```

XBasic

```
function Click as v ()
end function
```

dBASE

```
function nativeObject_Click()
return
```

event DblClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user double-clicks the left mouse button over an object.

Type	Description
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The DblClick event is fired when user double clicks the control. Use the ExpandOnDblClk property to specify whether an item is expanded or collapsed when user double clicks it.

Syntax for DblClick event, **/NET** version, on:

C#
```
private void DblClick(object sender,short Shift,int X,int Y)
{
}
```

VB
```
Private Sub DblClick(ByVal sender As System.Object,ByVal Shift As Short,ByVal X
As Integer,ByVal Y As Integer) Handles DblClick
End Sub
```

Syntax for DblClick event, **/COM** version, on:

C#
```
private void DblClick(object sender, AxEXPIVOTLib._IPivotEvents_DblClickEvent e)
{
}
```

C++
```
void OnDblClick(short Shift,long X,long Y)
{
}
```

C++ Builder
```
void __fastcall DblClick(TObject *Sender,short Shift,int X,int Y)
{
```

```
}
```

Delphi

```
procedure DblClick(ASender: TObject; Shift : Smallint;X : Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DblClick(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_DblClickEvent);
begin
end;
```

Powe…

```
begin event DblClick(integer Shift,long X,long Y)
end event DblClick
```

VB.NET

```
Private Sub DblClick(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_DblClickEvent) Handles DblClick
End Sub
```

VB6

```
Private Sub DblClick(Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub DblClick(ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Shift,X,Y
```

Xbas…

```
PROCEDURE OnDblClick(oPivot,Shift,X,Y)
RETURN
```

Syntax for DblClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="DblClick(Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DblClick(Shift,X,Y)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComDblClick Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
    Forward Send OnComDblClick llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_DblClick(Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DblClick(int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function DblClick as v (Shift as N,X as OLE::Exontrol.Pivot.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Pivot.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_DblClick(Shift,X,Y)
return
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

Type	Description
EventID as Long	A Long expression that specifies the identifier of the event. Each internal event of the control has an unique identifier. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties). The EventParam(-1) retrieves the number of parameters of fired event

The Event notification occurs ANY time the control fires an event. *This is useful for X++, which does not support event with parameters passed by reference. Also, this could be useful for C++ Builder or Delphi, which does not handle properly the events with parameters of VARIANT type.*

In X++ the "Error executing code: FormActiveXControl (data source), method ... called with invalid parameters" occurs when handling events that have parameters passed by reference. Passed by reference, means that in the event handler, you can change the value for that parameter, and so the control will takes the new value, and use it. The X++ is NOT able to handle properly events with parameters by reference, so we have the solution.

The solution is using and handling the Event notification and EventParam method., instead handling the event that gives the "invalid parameters" error executing code.

If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print expivot1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

Syntax for Event event, **/NET** version, on:

C#
```
private void Event(object sender,int EventID)
{
}
```

VB
```
Private Sub Event(ByVal sender As System.Object,ByVal EventID As Integer)
Handles Event
End Sub
```

Syntax for Event event, **/COM** version, on:

C#
```
private void Event(object sender, AxEXPIVOTLib._IPivotEvents_EventEvent e)
{
}
```

C++
```
void OnEvent(long EventID)
{
}
```

C++ Builder
```
void __fastcall Event(TObject *Sender,long EventID)
{
}
```

Delphi
```
procedure Event(ASender: TObject; EventID : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure Event(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_EventEvent);
begin
end;
```

Powe…

```
begin event Event(long EventID)
end event Event
```

VB.NET

```
Private Sub Event(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_EventEvent) Handles Event
End Sub
```

VB6

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VBA

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VFP

```
LPARAMETERS EventID
```

Xbas…

```
PROCEDURE OnEvent(oPivot,EventID)
RETURN
```

Syntax for Event event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Event(EventID)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Event(EventID)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComEvent Integer llEventID
   Forward Send OnComEvent llEventID
End_Procedure
```

Visual Objects

```
METHOD OCX_Event(EventID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Event(int _EventID)
{
```

```
}
```

XBasic

```
function Event as v (EventID as N)
end function
```

dBASE

```
function nativeObject_Event(EventID)
return
```

event KeyDown (KeyCode as Integer, Shift as Integer)

Occurs when the user presses a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key. You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the And operator with the shift argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

Syntax for KeyDown event, **/NET** version, on:

C#
```
private void KeyDown(object sender,ref short KeyCode,short Shift)
{
}
```

VB
```
Private Sub KeyDown(ByVal sender As System.Object,ByRef KeyCode As
Short,ByVal Shift As Short) Handles KeyDown
End Sub
```

Syntax for KeyDown event, **/COM** version, on:

C#
```
private void KeyDownEvent(object sender,
AxEXPIVOTLib._IPivotEvents_KeyDownEvent e)
```

```
{
}
```

C++

```
void OnKeyDown(short FAR* KeyCode,short Shift)
{
}
```

C++ Builder

```
void __fastcall KeyDown(TObject *Sender,short * KeyCode,short Shift)
{
}
```

Delphi

```
procedure KeyDown(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyDownEvent(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_KeyDownEvent);
begin
end;
```

Powe…

```
begin event KeyDown(integer KeyCode,integer Shift)
end event KeyDown
```

VB.NET

```
Private Sub KeyDownEvent(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_KeyDownEvent) Handles KeyDownEvent
End Sub
```

VB6

```
Private Sub KeyDown(KeyCode As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyDown(KeyCode As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift
```

Xbas…

```
PROCEDURE OnKeyDown(oPivot,KeyCode,Shift)
RETURN
```

Syntax for KeyDown event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function KeyDown(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComKeyDown Short llKeyCode Short llShift
   Forward Send OnComKeyDown llKeyCode llShift
End_Procedure
```

Visual Objects
```
METHOD OCX_KeyDown(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_KeyDown(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic
```
function KeyDown as v (KeyCode as N,Shift as N)
end function
```

dBASE
```
function nativeObject_KeyDown(KeyCode,Shift)
return
```

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	An integer that returns a standard numeric ANSI keycode

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress does not indicate the physical state of the keyboard; instead, it passes a character. KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters.

Syntax for KeyPress event, **/NET** version, on:

C#

```
private void KeyPress(object sender,ref short KeyAscii)
{
}
```

VB

```
Private Sub KeyPress(ByVal sender As System.Object,ByRef KeyAscii As Short)
Handles KeyPress
End Sub
```

Syntax for KeyPress event, **/COM** version, on:

C#

```
private void KeyPressEvent(object sender,
AxEXPIVOTLib._IPivotEvents_KeyPressEvent e)
{
}
```

C++

```
void OnKeyPress(short FAR* KeyAscii)
{
}
```

C++ Builder

```
void __fastcall KeyPress(TObject *Sender,short * KeyAscii)
{
}
```

Delphi

```
procedure KeyPress(ASender: TObject; var KeyAscii : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyPressEvent(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_KeyPressEvent);
begin
end;
```

Powe…

```
begin event KeyPress(integer KeyAscii)
end event KeyPress
```

VB.NET

```
Private Sub KeyPressEvent(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_KeyPressEvent) Handles KeyPressEvent
End Sub
```

VB6

```
Private Sub KeyPress(KeyAscii As Integer)
End Sub
```

VBA

```
Private Sub KeyPress(KeyAscii As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyAscii
```

Xbas…

```
PROCEDURE OnKeyPress(oPivot,KeyAscii)
RETURN
```

Syntax for KeyPress event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="KeyPress(KeyAscii)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyPress(KeyAscii)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyPress Short llKeyAscii
   Forward Send OnComKeyPress llKeyAscii
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii)
{
}
```

XBasic

```
function KeyPress as v (KeyAscii as N)
end function
```

dBASE

```
function nativeObject_KeyPress(KeyAscii)
return
```

event KeyUp (KeyCode as Integer, Shift as Integer)

Occurs when the user releases a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the KeyUp event procedure to respond to the releasing of a key.

Syntax for KeyUp event, **/NET** version, on:

C#

```
private void KeyUp(object sender,ref short KeyCode,short Shift)
{
}
```

VB

```
Private Sub KeyUp(ByVal sender As System.Object,ByRef KeyCode As Short,ByVal
Shift As Short) Handles KeyUp
End Sub
```

Syntax for KeyUp event, **/COM** version, on:

C#

```
private void KeyUpEvent(object sender, AxEXPIVOTLib._IPivotEvents_KeyUpEvent
e)
{
}
```

C++

```
void OnKeyUp(short FAR* KeyCode,short Shift)
{
}
```

C++ Builder

```
void __fastcall KeyUp(TObject *Sender,short * KeyCode,short Shift)
{
```

```
}
```

Delphi

```
procedure KeyUp(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyUpEvent(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_KeyUpEvent);
begin
end;
```

Powe…

```
begin event KeyUp(integer KeyCode,integer Shift)
end event KeyUp
```

VB.NET

```
Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_KeyUpEvent) Handles KeyUpEvent
End Sub
```

VB6

```
Private Sub KeyUp(KeyCode As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyUp(KeyCode As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift
```

Xbas…

```
PROCEDURE OnKeyUp(oPivot,KeyCode,Shift)
RETURN
```

Syntax for KeyUp event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="KeyUp(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyUp(KeyCode,Shift)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyUp Short llKeyCode Short llShift
    Forward Send OnComKeyUp llKeyCode llShift
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyUp(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyUp(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic

```
function KeyUp as v (KeyCode as N,Shift as N)
end function
```

dBASE

```
function nativeObject_KeyUp(KeyCode,Shift)
return
```

event LayoutEndChanging (Operation as LayoutChangingEnum)

Notifies your application once the control's layout has been changed.

Type	Description
Operation as LayoutChangingEnum	A LayoutChangingEnum expression that indicates the operation that ends.

The LayoutEndChanging event notifies your application once the current operation ends. The LayoutStartChanging event notifies your application once the user/control is about to being an operation. Use the PivotBarVisible property to enable or disable different properties of the control's pivot bar like: AutoHide, Undo/Redo support, and so on.

These events notify your application if any of the following operations occur:

- The user resizes one of the control's panels
- The pivot bar header is shown or hidden if the AutoHide option is On.
- The user drags any column to the pivot bar
- The user removes by dragging a column from the pivot bar
- The control organizes the current layout (sort and group the column, executes the aggregate functions, and so on)
- The user presses CTRL + Z or CTRL + Y to do an Undo/Redo operation

Syntax for LayoutEndChanging event, **/NET** version, on:

C#

```
private void LayoutEndChanging(object
sender,exontrol.EXPIVOTLib.LayoutChangingEnum Operation)
{
}
```

VB

```
Private Sub LayoutEndChanging(ByVal sender As System.Object,ByVal Operation
As exontrol.EXPIVOTLib.LayoutChangingEnum) Handles LayoutEndChanging
End Sub
```

Syntax for LayoutEndChanging event, **/COM** version, on:

C#

```
private void LayoutEndChanging(object sender,
AxEXPIVOTLib._IPivotEvents_LayoutEndChangingEvent e)
{
}
```

C++

```
void OnLayoutEndChanging(long Operation)
```

```
{
}
```

C++ Builder

```
void __fastcall LayoutEndChanging(TObject
*Sender,Expivotlib_tlb::LayoutChangingEnum Operation)
{
}
```

Delphi

```
procedure LayoutEndChanging(ASender: TObject; Operation :
LayoutChangingEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure LayoutEndChanging(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_LayoutEndChangingEvent);
begin
end;
```

Powe…

```
begin event LayoutEndChanging(long Operation)
end event LayoutEndChanging
```

VB.NET

```
Private Sub LayoutEndChanging(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_LayoutEndChangingEvent) Handles
LayoutEndChanging
End Sub
```

VB6

```
Private Sub LayoutEndChanging(ByVal Operation As
EXPIVOTLibCtl.LayoutChangingEnum)
End Sub
```

VBA

```
Private Sub LayoutEndChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation
```

Xbas…

```
PROCEDURE OnLayoutEndChanging(oPivot,Operation)
RETURN
```

Syntax for LayoutEndChanging event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="LayoutEndChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function LayoutEndChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComLayoutEndChanging OLELayoutChangingEnum llOperation
    Forward Send OnComLayoutEndChanging llOperation
End_Procedure
```

Visual Objects
```
METHOD OCX_LayoutEndChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_LayoutEndChanging(int _Operation)
{
}
```

XBasic
```
function LayoutEndChanging as v (Operation as
OLE::Exontrol.Pivot.1::LayoutChangingEnum)
end function
```

dBASE
```
function nativeObject_LayoutEndChanging(Operation)
return
```

For instance, you can use the LayoutStartChaging/LayoutEndChaging event to notify your application once the user drag and drop columns/aggregate functions to the control's pivot bar. By default, the control is keeping the layout (size, position, sorting order, …) of the generated columns when the user makes changes in the control's pivot bar. In other words, if you include a new value in the column's filter, the new generated columns will be appended at the end of the header list. In order to prevent this, you need to handle the LayoutEndChanging event when exPivotDataColumnSort and exPivotDataColumnFilterChange notifications occur, and call the control's Reset method as in the following sample:

```
Private Sub Pivot1_LayoutEndChanging(ByVal Operation As
EXPIVOTLibCtl.LayoutChangingEnum)
   If (Operation = exPivotDataColumnSort) Or (Operation =
exPivotDataColumnFilterChange) Then
      Pivot1.Reset "c*.position*"
   End If
End Sub
```

The sample resets the position of the generated columns when exPivotDataColumnSort and exPivotDataColumnFilterChange notifications occur.

event LayoutStartChanging (Operation as LayoutChangingEnum)

Occurs when the control's layout is about to be changed.

Type	Description
Operation as LayoutChangingEnum	A LayoutChangingEnum expression that indicates the operation to be started.

The LayoutStartChanging event notifies your application once the user/control is about to being an operation. The LayoutEndChanging event notifies your application once the current operation ends. Use the PivotBarVisible property to enable or disable different properties of the control's pivot bar like: AutoHide, Undo/Redo support, and so on.

These events notify your application if any of the following operations occur:

- The user resizes one of the control's panels
- The pivot bar header is shown or hidden if the AutoHide option is On.
- The user drags any column to the pivot bar
- The user removes by dragging a column from the pivot bar
- The control organizes the current layout (sort and group the column, executes the aggregate functions, and so on)
- The user presses CTRL + Z or CTRL + Y to do an Undo/Redo operation

Syntax for LayoutStartChanging event, **/NET** version, on:

C#

```
private void LayoutStartChanging(object
sender,exontrol.EXPIVOTLib.LayoutChangingEnum Operation)
{
}
```

VB

```
Private Sub LayoutStartChanging(ByVal sender As System.Object,ByVal Operation
As exontrol.EXPIVOTLib.LayoutChangingEnum) Handles LayoutStartChanging
End Sub
```

Syntax for LayoutStartChanging event, **/COM** version, on:

C#

```
private void LayoutStartChanging(object sender,
AxEXPIVOTLib._IPivotEvents_LayoutStartChangingEvent e)
{
}
```

C++

```
void OnLayoutStartChanging(long Operation)
```

```
{
}
```

C++ Builder

```
void __fastcall LayoutStartChanging(TObject
*Sender,Expivotlib_tlb::LayoutChangingEnum Operation)
{
}
```

Delphi

```
procedure LayoutStartChanging(ASender: TObject; Operation :
LayoutChangingEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure LayoutStartChanging(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_LayoutStartChangingEvent);
begin
end;
```

Powe…

```
begin event LayoutStartChanging(long Operation)
end event LayoutStartChanging
```

VB.NET

```
Private Sub LayoutStartChanging(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_LayoutStartChangingEvent) Handles
LayoutStartChanging
End Sub
```

VB6

```
Private Sub LayoutStartChanging(ByVal Operation As
EXPIVOTLibCtl.LayoutChangingEnum)
End Sub
```

VBA

```
Private Sub LayoutStartChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation
```

Xbas…

```
PROCEDURE OnLayoutStartChanging(oPivot,Operation)
RETURN
```

Syntax for LayoutStartChanging event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="LayoutStartChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function LayoutStartChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComLayoutStartChanging OLELayoutChangingEnum llOperation
    Forward Send OnComLayoutStartChanging llOperation
End_Procedure
```

Visual Objects
```
METHOD OCX_LayoutStartChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_LayoutStartChanging(int _Operation)
{
}
```

XBasic
```
function LayoutStartChanging as v (Operation as
OLE::Exontrol.Pivot.1::LayoutChangingEnum)
end function
```

dBASE
```
function nativeObject_LayoutStartChanging(Operation)
return
```

For instance, you can use the LayoutStartChaging/LayoutEndChaging event to notify your application once the user drag and drop columns/aggregate functions to the control's pivot bar.

event MouseDown (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user presses a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. The AnchorClick event notifies your application that the user clicks an <a> anchor element.

Syntax for MouseDown event, **/NET** version, on:

C#

```
private void MouseDownEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseDownEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseDownEvent
End Sub
```

Syntax for MouseDown event, **/COM** version, on:

C#

```
private void MouseDownEvent(object sender,
AxEXPIVOTLib._IPivotEvents_MouseDownEvent e)
{
```

```
}
```

C++

```
void OnMouseDown(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall MouseDown(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure MouseDown(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseDownEvent(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_MouseDownEvent);
begin
end;
```

Powe…

```
begin event MouseDown(integer Button,integer Shift,long X,long Y)
end event MouseDown
```

VB.NET

```
Private Sub MouseDownEvent(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_MouseDownEvent) Handles MouseDownEvent
End Sub
```

VB6

```
Private Sub MouseDown(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub MouseDown(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseDown(oPivot,Button,Shift,X,Y)
```

```
RETURN
```

Syntax for MouseDown event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseDown(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseDown Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseDown llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseDown(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseDown(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseDown as v (Button as N,Shift as N,X as
OLE::Exontrol.Pivot.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Pivot.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseDown(Button,Shift,X,Y)
return
```

event MouseMove (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user moves the mouse.

Type	Description
Button as Integer	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. The Background(exCursorHoverColumn) property specifies the visual appearance of the column's header when the cursor hovers it. Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the point.

Syntax for MouseMove event, **/NET** version, on:

C#

```
private void MouseMoveEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseMoveEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseMoveEvent
End Sub
```

Syntax for MouseMove event, **/COM** version, on:

C#

```
private void MouseMoveEvent(object sender,
AxEXPIVOTLib._IPivotEvents_MouseMoveEvent e)
{
```

```
}
```

C++

```
void OnMouseMove(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall MouseMove(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure MouseMove(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseMoveEvent(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_MouseMoveEvent);
begin
end;
```

Powe…

```
begin event MouseMove(integer Button,integer Shift,long X,long Y)
end event MouseMove
```

VB.NET

```
Private Sub MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_MouseMoveEvent) Handles MouseMoveEvent
End Sub
```

VB6

```
Private Sub MouseMove(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub MouseMove(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseMove(oPivot,Button,Shift,X,Y)
```

```
RETURN
```

Syntax for MouseMove event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseMove(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseMove(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseMove Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseMove llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseMove(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseMove as v (Button as N,Shift as N,X as
OLE::Exontrol.Pivot.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Pivot.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseMove(Button,Shift,X,Y)
return
```

event MouseUp (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user releases a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the point. The AnchorClick event notifies your application that the user clicks an <a> anchor element.

Syntax for MouseUp event, **/NET** version, on:

C#
```
private void MouseUpEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB
```
Private Sub MouseUpEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseUpEvent
End Sub
```

Syntax for MouseUp event, **/COM** version, on:

C#
```
private void MouseUpEvent(object sender,
AxEXPIVOTLib._IPivotEvents_MouseUpEvent e)
```

```
{
}
```

C++

```
void OnMouseUp(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall MouseUp(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure MouseUp(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseUpEvent(sender: System.Object; e:
AxEXPIVOTLib._IPivotEvents_MouseUpEvent);
begin
end;
```

Powe…

```
begin event MouseUp(integer Button,integer Shift,long X,long Y)
end event MouseUp
```

VB.NET

```
Private Sub MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXPIVOTLib._IPivotEvents_MouseUpEvent) Handles MouseUpEvent
End Sub
```

VB6

```
Private Sub MouseUp(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub MouseUp(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseUp(oPivot,Button,Shift,X,Y)
RETURN
```

Syntax for MouseUp event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseUp(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseUp(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseUp Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseUp llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseUp(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseUp(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseUp as v (Button as N,Shift as N,X as
OLE::Exontrol.Pivot.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Pivot.1::OLE_YPOS_PIXELS)
end function
```

dBASE

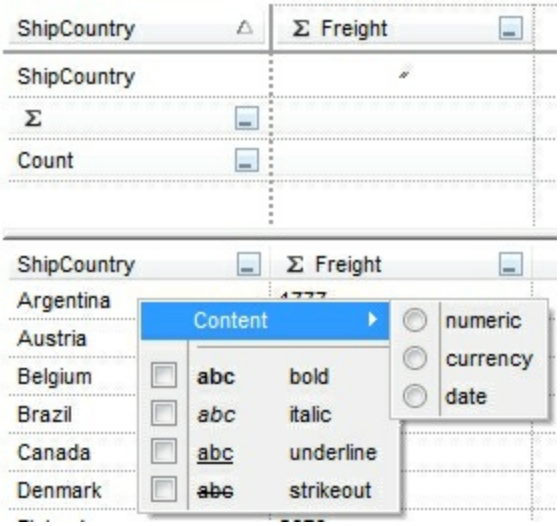
```
function nativeObject_MouseUp(Button,Shift,X,Y)
return
```

event RClick ()

Fired when right mouse button is clicked

Type	Description

The RClick event notifies your application when the user right clicks the control. Use the Click event to notify your application that the user clicks the control (using the left mouse button). Use the MouseDown or MouseUp event if you require the cursor position during the RClick event. By default, the control's context menu is shown when user right-clicks the control. The context menu displays the attributes or format to be changed for the element from the cursor as shown in the following screen shot:



You can use the PivotBarVisible property to disable the default context menu (exPivotBarAllowFormatAppearance, exPivotBarAllowFormatContent). Use the FormatAppearance.Add method to add more appearance functions to the control's context menu, or FormatContents.Add method to add format functions to the control's context menu.

Syntax for RClick event, **/NET** version, on:

C#
```
private void RClick(object sender)
{
}
```

VB
```
Private Sub RClick(ByVal sender As System.Object) Handles RClick
End Sub
```

Syntax for RClick event, **/COM** version, on:

C#
```
private void RClick(object sender, EventArgs e)
```

```
{
}
```

C++

```
void OnRClick()
{
}
```

C++ Builder

```
void __fastcall RClick(TObject *Sender)
{
}
```

Delphi

```
procedure RClick(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure RClick(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event RClick()
end event RClick
```

VB.NET

```
Private Sub RClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles RClick
End Sub
```

VB6

```
Private Sub RClick()
End Sub
```

VBA

```
Private Sub RClick()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnRClick(oPivot)
RETURN
```

Syntax for RClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="RClick()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function RClick()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComRClick
   Forward Send OnComRClick
End_Procedure
```

Visual Objects

```
METHOD OCX_RClick() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_RClick()
{
}
```

XBasic

```
function RClick as v ()
end function
```

dBASE

```
function nativeObject_RClick()
return
```

Expressions

An expression is a string which defines a formula or criteria, that's evaluated at runtime. The expression may be a combination of variables, constants, strings, dates and operators/functions. For instance 1000 format `` gets 1,000.00 for US format, while 1.000,00 is displayed for German format.

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

Usage examples:

- 100 + 200, adds numbers and returns 300
- "100" + 200, concatenates the strings, and returns "100200"
- currency(1000) displays the value in currency format based on the current regional setting, such as "$1,000.00" for US format.
- 1000 format `` gets 1,000.00 for English format, while 1.000,00 is displayed for German format
- 1000 format `2|.|3|,` always gets 1,000.00 no matter of settings in the control panel.
- upper("string") converts the giving string in uppercase letters, such as "STRING"
- date(dateS('3/1/' + year(9:=#1/1/2018#)) + ((1:=(((255 - 11 * (year(=:9) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(=:9) + int(year(=:9) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7)) returns the date the Easter Sunday will fall, for year 2018. In this case the expression returns #4/1/2018#. If #1/1/2018# is replaced with #1/1/2019#, the expression returns #4/21/2019#.

Listed bellow are all predefined constants, operators and functions the general-expression supports:

The constants can be represented as:

- numbers in **decimal** format (where dot character specifies the decimal separator). For instance: -1, 100, 20.45, .99 and so on
- numbers in **hexa-decimal** format (preceded by **0x** or **0X** sequence), uses sixteen distinct symbols, most often the symbols 0-9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. Hexadecimal numerals are widely used by computer system designers and programmers. As each hexadecimal digit represents four binary digits (bits), it allows a more human-friendly representation of binary-coded values. For instance, 0xFF,

0x00FF00, and so so.

- **date-time** in format **#mm/dd/yyyy hh:mm:ss#**, For instance, #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM
- **string**, if it starts / ends with any of the **'** or ` or **"** characters. If you require the starting character inside the string, it should be escaped ( preceded by a \ character ). For instance, `Mihai`, "Filimon", 'has', "\"a quote\"", and so on

The predefined constants are:

- **bias** (BIAS constant), defines the difference, in minutes, between Coordinated Universal Time (UTC) and local time. For example, Middle European Time (MET, GMT+01:00) has a time zone bias of "-60" because it is one hour ahead of UTC. Pacific Standard Time (PST, GMT-08:00) has a time zone bias of "+480" because it is eight hours behind UTC. For instance, date(value - bias/24/60) converts the UTC time to local time, or date(date('now') + bias/24/60) converts the current local time to UTC time. For instance, "date(value - bias/24/60)" converts the value date-time from UTC to local time, while "date(value + bias/24/60)" converts the local-time to UTC time.
- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- **/** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported binary range operators, all these with the same priority 5, are :

- a **MIN** b (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- a **MAX** b (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

The supported binary operators, all these with the same priority 0, are :

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For

instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for *?* operator is

  expression ? true_part : false_part

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

  expression array (c1,c2,c3,...cn)

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')* is equivalent with *month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'O';10:'N';11:'D')*.

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  expression in (c1,c2,c3,...cn)

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *value in (11,22,33,44,13)* is equivalent with *(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or*

statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

expression switch (default,c1,c2,c3,...,cn)

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the *%0 switch ('not found',1,4,7,9,11)* gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the *date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)* indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: *date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)* statement indicates the working hours for dates as follows:

  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
  - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions. Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. The type operator may return any of the following: 0 - empty (not initialized), 1 - null, 2 - short, 3 - long, 4 - float, 5 - double, 6 - currency, **7 - date**, **8 - string**, 9 - object, 10 - error, **11 - boolean**, 12 - variant, 13 - any, 14 - decimal, 16 - char, 17 - byte, 18 - unsigned short, 19 - unsigned long, 20 - long on 64 bits, 21 - unsigned long on 64 bites. For instance *type(%1) = 8* specifies the cells (on the column with the index 1) that contains string values.
- **str** (unary operator) converts the expression to a string. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#
- **hex** (unary operator) converts the giving string from hexa-representation to a numeric value, or converts the giving numeric value to hexa-representation as string. For instance, hex(`FF`) returns 255, while the hex(255) or hex(0xFF) returns the `FF` string. The hex(hex(`FFFFFFFF`)) always returns `FFFFFFFF` string, as the second hex call converts the giving string to a number, and the first hex call converts the returned number to string representation (hexa-representation).

The bitwise operators for numbers are:

- a **bitand** b (binary operator) computes the AND operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitand 0x10111000 returns 0x00001000.
- a **bitor** b (binary operator) computes the OR operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitor 0x10111000 returns 0x11111000.
- a **bitxor** b (binary operator) computes the XOR (exclusive-OR) operation on bits of a and b, and returns the unsigned value. For instance, 0x01110010 bitxor 0x10101010 returns 0x11011000.
- a **bitshift** (b) (binary operator) shifts every bit of a value to the left if b is negative, or to the right if b is positive, for b times, and returns the unsigned value. For instance, 128 bitshift 1 returns 64 (dividing by 2) or 128 bitshift (-1) returns 256 (multiplying by

2)

- **bitnot** (unary operator) flips every bit of x, and returns the unsigned value. For instance, bitnot(0x00FF0000) returns 0xFF00FFFF.

The operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the *1000 format ''* displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of

the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.

- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

The operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- a **startwith** b (binary operator) specifies whether a string starts with specified string (

0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1

- a **endwith** b (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- a **contains** b (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- a **left** b (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- a **right** b (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- a **split** b (binary operator) splits the a using the separator b, and returns an array. For instance, the *weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' split ' '* gets the weekday as string. This operator can be used with the array.
- a **like** b (binary operator) compares the string a against the pattern b. The pattern b may contain wild-characters such as *, ?, # or [] and can have multiple patterns separated by space character. In order to have the space, or any other wild-character inside the pattern, it has to be escaped, or in other words it should be preceded by a \ character. For instance value like `F*e` matches all strings that start with F and ends on e, or value like `a* b*` indicates any strings that start with a or b character.
- a **lpad** b (binary operator) pads the value of a to the left with b padding pattern. For instance, 12 lpad "0000" generates the string "0012".
- a **rpad** b (binary operator) pads the value of a to the right with b padding pattern. For instance, 12 lpad "____" generates the string "12__".
- a **concat** b (binary operator) concatenates the a (as string) for b times. For instance, "x" concat 5, generates the string "xxxxx".

The operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"

- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The expression supports also **immediate if** (similar with iif in visual basic, or ? : in C++) ie cond ? value_true : value_false, which means that once that cond is true the value_true is used, else the value_false is used. Also, it supports variables, up to 10 from 0 to 9. For instance, 0:="Abc" means that in the variable 0 is "Abc", and =:0 means retrieves the value of the variable 0. For instance, the len(%0) ? (0:=(%1+%2) ? currency(=:0) else `` ) : `` gets the sum between second and third column in currency format if it is not zero, and only if the first column is not empty. As you can see you can use the variables to avoid computing several times the same thing (in this case the sum %1 and %2 .

		Sum of Freight		
ShipCountry ShipCity	Sum of Freight	Federal Shipping	Speedy Express	United Package
Total	**$202,953.00**	**$63,330.00**	**$52,168.00**	**$87,455.00**
Argentina	**1,777.00**	**111.00**	**253.00**	**1,413.00**
Buenos Aires	1,777.00	111.00	253.00	1,413.00
Austria	**27,730.00**	**6,565.00**	**9,065.00**	**12,100.00**
Graz	24,540.00	5,616.00	8,405.00	10,519.00
Salzburg	3,190.00	949.00	660.00	1,581.00
Belgium	**4,274.00**	**1,099.00**	**765.00**	**2,410.00**
Bruxelles	1,087.00	76.00	558.00	453.00
Charleroi	3,187.00	1,023.00	207.00	1,957.00
Brazil	**14,586.00**	**2,227.00**	**4,823.00**	**7,536.00**
Campinas	883.00	687.00	72.00	124.00
Resende	547.00	144.00	359.00	44.00
Rio de Janeiro	4,313.00	383.00	2,166.00	1,764.00
São Paulo	8,843.00	1,013.00	2,226.00	5,604.00
Canada	**6,324.00**	**2,151.00**	**725.00**	**3,448.00**
Montréal	4,124.00	673.00	465.00	2,986.00
Tsawassen	2,171.00	1,449.00	260.00	462.00
Vancouver	29.00	29.00		
Denmark	**4,283.00**	**1,373.00**	**883.00**	**2,027.00**
Křbenhavn	1,498.00	418.00	34.00	1,046.00
Ĺrhus	2,785.00	955.00	849.00	981.00
Finland	**2,873.00**	**549.00**	**657.00**	**1,667.00**
Helsinki	263.00	139.00	124.00	
Oulu	2,610.00	410.00	533.00	1,667.00
France	**12,435.00**	**6,066.00**	**2,908.00**	**3,461.00**
Lille	2,498.00	2,059.00	33.00	406.00
Lyon	1,566.00	1,114.00	270.00	182.00
Marseille	3,954.00	1,711.00	1,041.00	1,202.00
Nantes	409.00	88.00	193.00	128.00
Paris	282.00	9.00	3.00	270.00
Reims	147.00	118.00	12.00	17.00
Strasbourg	1,980.00	718.00	463.00	799.00
Toulouse	1,335.00	249.00	674.00	412.00
Versailles	264.00		219.00	45.00
Germany	**34,461.00**	**11,829.00**	**12,673.00**	**9,959.00**
Aachen	595.00	396.00	158.00	41.00
Berlin	418.00	140.00	217.00	61.00

Brandenburg	3,037.00	1,212.00	87.00	1,738.00
Cunewalde	17,244.00	7,302.00	6,571.00	3,371.00
Frankfurt a.M.	2,846.00	32.00	1,444.00	1,370.00
Köln	2,671.00	712.00	1,132.00	827.00
Leipzig	897.00	253.00	528.00	116.00
Mannheim	353.00	108.00	0.00	245.00
München	4,928.00	1,674.00	2,004.00	1,250.00
Münster	262.00		84.00	178.00
Stuttgart	1,210.00		448.00	762.00
Ireland	**7,214.00**	**1,841.00**	**793.00**	**4,580.00**
Cork	7,214.00	1,841.00	793.00	4,580.00
Italy	**2,106.00**	**501.00**	**1,195.00**	**410.00**
Bergamo	1,205.00	483.00	522.00	200.00
Reggio Emilia	755.00	18.00	549.00	188.00
Torino	146.00		124.00	22.00
Mexico	**3,119.00**	**1,114.00**	**497.00**	**1,508.00**
México D.F.	3,119.00	1,114.00	497.00	1,508.00
Norway	**898.00**	**39.00**	**26.00**	**833.00**
Stavern	898.00	39.00	26.00	833.00
Poland	**462.00**	**130.00**	**9.00**	**323.00**
Warszawa	462.00	130.00	9.00	323.00
Portugal	**1,721.00**	**596.00**	**134.00**	**991.00**
Lisboa	1,721.00	596.00	134.00	991.00
Spain	**2,416.00**	**287.00**	**517.00**	**1,612.00**
Barcelona	68.00	8.00	40.00	20.00
Madrid	651.00	24.00	183.00	444.00
Sevilla	1,697.00	255.00	294.00	1,148.00
Sweden	**10,141.00**	**1,599.00**	**3,159.00**	**5,383.00**
Bräcke	5,304.00	863.00	2,615.00	1,826.00
Luleĺ	4,837.00	736.00	544.00	3,557.00
Switzerland	**3,911.00**	**2,100.00**	**766.00**	**1,045.00**
Bern	939.00	400.00	371.00	168.00
Genčve	2,972.00	1,700.00	395.00	877.00
UK	**8,486.00**	**2,551.00**	**2,434.00**	**3,501.00**
Colchester	1,448.00	189.00	84.00	1,175.00
Cowes	1,144.00	140.00	624.00	380.00
London	5,894.00	2,222.00	1,726.00	1,946.00
USA	**45,861.00**	**18,372.00**	**7,742.00**	**19,747.00**
Albuquerque	6,493.00	4,421.00	692.00	1,380.00

Anchorage	2,554.00	1,586.00	885.00	83.00
Boise	26,534.00	8,344.00	5,371.00	12,819.00
Butte	370.00	45.00	40.00	285.00
Elgin	301.00	60.00	201.00	40.00
Eugene	2,456.00	261.00	171.00	2,024.00
Kirkland	241.00	14.00		227.00
Lander	1,654.00	13.00	83.00	1,558.00
Portland	677.00	479.00	39.00	159.00
San Francisco	546.00			546.00
Seattle	4,016.00	3,142.00	260.00	614.00
Walla Walla	19.00	7.00		12.00
Venezuela	**7,875.00**	**2,230.00**	**2,144.00**	**3,501.00**
Barquisimeto	2,217.00	901.00	138.00	1,178.00
Caracas	136.00	136.00		
I. de Margarita	2,104.00	624.00	898.00	582.00
San Cristóbal	3,418.00	569.00	1,108.00	1,741.00