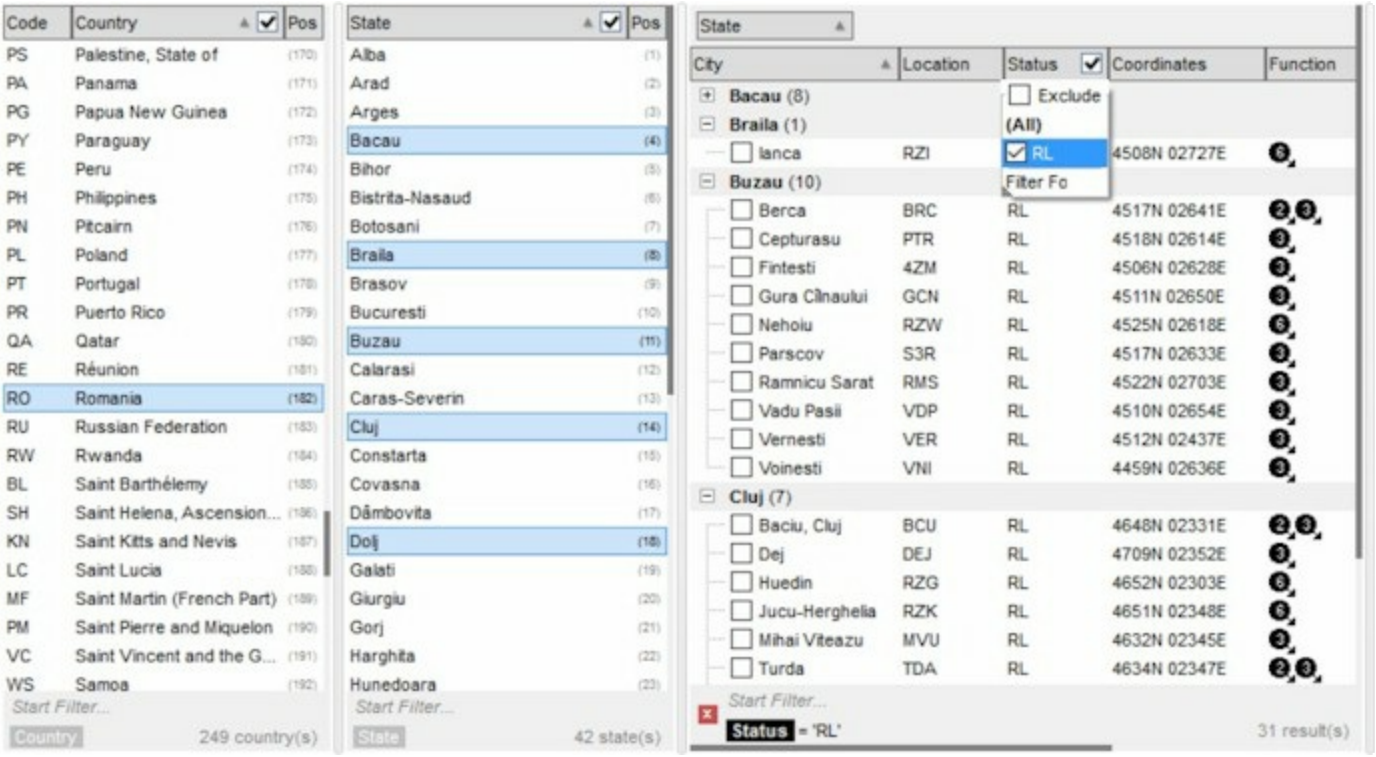
ExCascadeTree

The eXCascadeTree component is a multiple-columns-tree-view component that uses miller columns visualization to display your data. The Miller columns (also known as Cascading Lists) are a browsing/visualization technique that can be applied to tree structures. The cascade columns allow multiple levels of the hierarchy to be open at once, and provide a visual representation of the current location. It is closely related to techniques used earlier in the Smalltalk browser, but was independently invented by Mark S. Miller in 1980 at Yale University.

Features include:

- Array, ADO, DAO, XML, DataSet, Multiple-Data Source support
- Split View support
- Group By support
- Conditional Format support
- Total Fields support (Aggregate functions: sum, min, max, count, avg)
- Incremental Search support
- FilterBar support
- StatusBar support
- Customizable Context Menu support
- ScrollBar Extension support



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

Specifies the object's alignment.

Name	Value	Description
LeftAlignment	0	The source is left aligned.
CenterAlignment	1	The source is centered.
RightAlignment	2	The source is right aligned.

constants AllowSplitViewEnum

The AllowSplitViewEnum type specifies how many vertically split-panels the control support. The AllowSplitView property specifies whether the user can split the control into multiple-views. The AllowSplitViewEnum type supports the following values:

Name	Value	Description
exNoSplitView	0	No vertically split-view is allowed.
exAllowOneSplitView	1	One additional vertically split-panel is allowed.
exAllowTwoSplitView	2	Two additional vertically split-panel are allowed.

constants AppearanceEnum

The AppearanceEnum enumeration is used to specify the appearance of the control's header bar.

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	The item can be dragged from a position to another, but not outside of its group. If your items are arranged as a flat list, no hierarchy, this option can be used to allow the user change the item's position at runtime by drag and drop. This option does not change the parent of any dragged item. The dragging items could be the focused item or a contiguously selection. Click the selection and moves the cursor up or down, so the position of the dragging items is changed. The draggable collection is a collection of sortable items between 2 non-sortable items (SortableItem property). The drag and drop operation can not start on a non-sortable or non-selectable item (SelectableItem property). In other words, you can specify a range where an item can be dragged using the SortableItem property. Just set the SortableItem property on False, for margins, and so the items can be dragged between these items only.
		The item can be dragged to any position or to any parent, while the dragging object keeps its indentation. This option can be used to allow the user change the item's position at runtime by drag and drop. In the same time, the parent's item could

exAutoDragPositionKeepIndent	2	be changed but keeping the item's indentation. The dragging items could be the focused item or a contiguously selection. Click the selection and moves the cursor up or down, so the position or parent of the dragging items is changed. The drag and drop operation can not start on a non-sortable or non-selectable item (SelectableItem property). In other words, you can specify a range where an item can be dragged using the SortableItem property. Just set the SortableItem property on False, for margins, and so the items can be dragged between these items only.
exAutoDragPositionAny	3	The item can be dragged to any position or to any parent, with no restriction. The dragging items could be the focused item or a contiguously selection. The parent of the dragging items could change with no restrictions, based on the position of the dragging item. Click the selection and moves the cursor up or down, so the position or parent of the dragging items is changed. Click the selection and moves the cursor left or right, so the item's indentation is decreased or increased. The drag and drop operation can not start on a non-sortable or non-selectable item (SelectableItem property). In other words, you can specify a range where an item can be dragged using the SortableItem property. Just set the SortableItem property on False, for margins, and so the items can be dragged between these items only. Click here to watch a movie on how exAutoDragCopyText works.
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.
		Drag and drop the selected items to a target application, and paste them as text only. Ability to

exAutoDragCopyText	9	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.
exAutoDragCopyImage	10	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 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. Use the ScrollBySingleLine property on False, to allow scrolling pixel by pixel when user clicks the up or down buttons on the vertical scroll bar. Use the ContinueColumnScroll property on True to allow scrolling the columns pixel by pixel. Click here or 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	2560	exAutoDragCopyImageOnShortTouch. Drag and 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	67108864	exAutoDragPositionAnyOnLongTouch. The object 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	268435456	exAutoDragCopyTextOnLongTouch. Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnLongTouch	536870912	exAutoDragCopyImageOnLongTouch. Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnLongTouch	1073741824	exAutoDragCopySnapShotOnLongTouch. Drag and drop a snap shot of the current component.
exAutoDragScrollOnLongTouch	2684354560	exAutoDragScrollOnLongTouch. The component is scrolled by clicking the object and dragging to a new position.

constants AutoSearchEnum

Specifies the kind of searching while user types characters within a column. Use the AutoSearch property to allow 'start with' incremental search or 'contains' incremental search feature in the control.

Name	Value	Description
exStartWith	0	Defines the 'starts with' incremental search within the column. If the user type characters within the column the control looks for items that start with the typed characters.
exContains	1	Defines the 'contains' incremental search within the column. If the user type characters within the column the control looks for items that contain the typed characters.

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.

Name	Value	Description
exSplitBar	18	Specifies the visual appearance for control's split bar.
exHeaderFilterBarActive	41	exHeaderFilterBarActive. Specifies the visual appearance of the drop down filter bar button, while filter is applied to the column.
exToolTipAppearance	64	Specifies the visual appearance of the borders of the tooltips.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.
exHSplitBar	141	Specifies the visual appearance for horizontal split bar.
exCSplitBar	142	Specifies the solid color / visual appearance of the split bar that creates new views.
exSelBackColorHide	166	exSelBackColorHide. Specifies the selection's background color, when the control has no focus.
exSelForeColorHide	167	exSelForeColorHide. Specifies the selection's foreground color, when the control has no focus.
exStatusBackColor	168	Specifies the status bar's background color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.
exStatusForeColor	169	Specifies the status bar's foreground color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.
		Specifies the size of the control's split bar, when

exSplitBarSize	170	resizing the cascade columns is enabled.
exDisableSplitBar	171	Specifies the visual appearance for control's split bar, when resizing the cascade columns is disabled.
exDisableSplitBarSize	172	Specifies the size of the control's split bar, when resizing the cascade columns is disabled.
exFocusFrame	173	Specifies the visual appearance of the frame around the focusing cascade column.
exStatusPanelBackColor	174	Specifies the status panel's background color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.
exTreeGlyphOpen	180	Specifies the visual appearance for the +/- buttons when it is collapsed.
exTreeGlyphClose	181	Specifies the visual appearance for the +/- buttons when it is expanded.
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.
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.
		The lower part (exLowerBackPart) in normal

exVSLower	268	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 .
		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

exScrollHoverAll	500	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 the background mode when painting the selected items. Use the SelBackMode property to specify the control's selection back mode.

Name	Value	Description
exOpaque	0	The selection is opaque.
exTransparent	1	The selection is transparent.
exGrid	2	The selection is half transparent half opaque

constants CascadeModeEnum

The CascadeModeEnum type specifies the modes the control supports. The Mode property indicates the mode the control displays the cascade columns. The CascadeModeEnum type supports the following values:

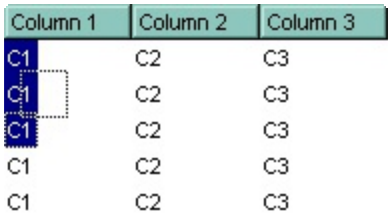
Name	Value	Description
exFixCascadeMode	0	Each cascade column can be displayed with a different width. The DefColumnWidth property specifies the width to create a new cascade column.
exSingleCascadeMode	1	No cascade columns support.
exSplitEqualCascadeMode	2	The cascade column fits equally the control's client area. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit.
exSplitFixCascadeMode	3	The cascade column fits equally the control's client area. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit.
exDisableResizeCascadeColumns	256	The user can't resize the cascade columns.
exAutoFitOnResizeClient	512	Each cascade column gets resized as soon as the control gets resized.

constants CellSelectEnum

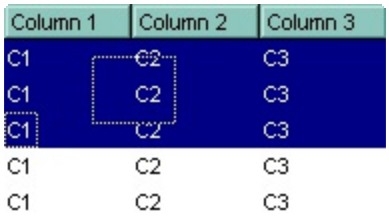
Specifies how the control selects cells or items within the control. Use the FullRowSelect property to enables full-row selection.

Name	Value	Description
exColumnSel	0	(**False**) Enables single-cell selection in the control.
exItemSel	-1	(**True**) Enables full-row selection in the control.
exRectSel	1	Enables rectangle selection in the control.

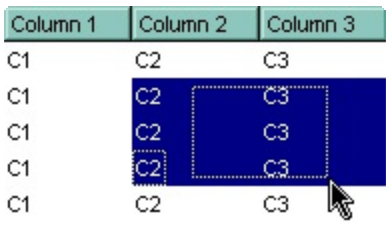
When the FullRowSelect property is **exColumnSel** the selection looks like:



When the FullRowSelect property is **exItemSel** the selection looks like:

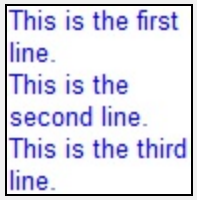
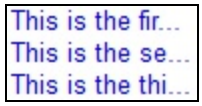


When the FullRowSelect property is **exRectSel** the selection looks like:



constants CellSingleLineEnum

The CellSingleLineEnum type defines whether the cell's caption is displayed on a single or multiple lines. The CellSingleLine property retrieves or sets a value indicating whether the cell is displayed using one line, or more than one line. The Def(exCellSingleLine) property specifies that all cells in the column display their content using multiple lines. The CellSingleLineEnum type supports the following values:

Name	Value	Description
exCaptionSingleLine	-1	Indicates that the cell's caption is displayed on a single line. In this case any \r\n or HTML tags is ignored. For instance the "This is the first line.\r\nThis is the second line.\r\nThis is the third line." shows as: 
exCaptionWordWrap	0	Specifies that the cell's caption is displayed on multiple lines, by wrapping the words. Any \r\n or HTML tag breaks the line. For instance the "This is the first line.\r\nThis is the second line.\r\nThis is the third line." shows as: 
exCaptionBreakWrap	1	Specifies that the cell's caption is displayed on multiple lines, by wrapping the breaks only. Only The \r\n or HTML tag breaks the line. For instance the "This is the first line.\r\nThis is the second line.\r\nThis is the third line." shows as: 

constants CheckStateEnum

Specifies the cell's state if CellHasCheckBox or CellHasRadioButton property is True.

Name	Value	Description
Unchecked	0	The cell is not checked.
Checked	1	The cell is checked.
PartialChecked	2	The cell is partially checked.

constants DefColumnEnum

The Def property retrieves or sets a value that indicates the default value of given properties for all cells in the same column.

Name	Value	Description
exCellHasCheckBox	0	Assigns check boxes to all cells in the column, if it is True. Similar with the CellHasCheckBox property. *(Boolean expression, False)*
exCellHasRadioButton	1	Assigns radio buttons to all cells in the column, if it is True. Similar with the CellHasRadioButton property. *(Boolean expression, False)*
exCellHasButton	2	Specifies that all cells in the column are buttons, if it is True. Similar with the CellHasButton property. *(Boolean expression, False)*
exCellButtonAutoWidth	3	Specifies that all buttons in the column fit the cell's caption, if it is True. Similar with the CellButtonAutoWidth property. *(Boolean expression, False)*
exCellBackColor	4	Specifies the background color for all cells in the column. Use the CellBackColor property to assign a background color for a specific cell. The property has effect only if the property is different than zero. *(Long expression)*
exCellForeColor	5	Specifies the foreground color for all cells in the column. Use the CellForeColor property to assign a foreground color for a specific cell. The property has effect only if the property is different than zero. *(Long expression)*

exCellVAlignment	6	Specifies the column's vertical alignment. By default, the Def(exCellVAlignment) property is exMiddle. Use the CellVAlignment property to specify the vertical alignment for a particular cell. *(VAlignmentEnum expression, exMiddle)*
exHeaderBackColor	7	Specifies the column's header background color. The property has effect only if the property is different than zero. Use this option to change the background color for a column in the header area. The exHeaderBackColor option supports skinning, so 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. *(Color expression)*
exHeaderForeColor	8	Specifies the column's header background color. The property has effect only if the property is different than zero. *(Color expression)*
exCellSingleLine	16	Specifies that all cells in the column displays its content into single or multiple lines. Similar with the CellSingleLine property. If using the CellSingleLine / Def(exCellSingleLine) property, we recommend to set the ScrollBySingleLine property on True so all items can be scrolled. *(CellSingleLineEnum type, previously Boolean expression)*
exCellValueFormat	17	Similar with the CellValueFormat property, *(ValueFormatEnum expression, exText)*
		Specifies the format layout for the cells. The CellFormatLevel property indicates the format layout for a specified cell. Use the FormatLevel

exCellFormatLevel	32	property to specify the layout of the column in the control's header bar. *(CRD string expression)*
exCellOwnerDraw	33	reserved
exCellDrawPartsOrder	34	Specifies the order of the drawing parts for the entire column. By default, this option is "check,icon,icons,picture,caption", which means that the cell displays its parts in the following order: check box/ radio buttons (CellHasCheckBox/CellRadioButton), single icon (CellImage), multiple icons (CellImages), custom size picture (CellPicture), and the cell's caption. Use the exCellDrawPartsOrder option to specify a new order for the drawing parts in the cells of the column. The RightToLeft property automatically flips the order of the columns. *(String expression, "check,icon,icons,picture,caption")*
exCellPaddingLeft	48	Gets or sets the left padding (space) of the cells within the column. *(Long expression)*
exCellPaddingRight	49	Gets or sets the right padding (space) of the cells within the column. *(Long expression)*
exCellPaddingTop	50	Gets or sets the top padding (space) of the cells within the column. *(Long expression)*
exCellPaddingBottom	51	Gets or sets the bottom padding (space) of the cells within the column. *(Long expression)*

exHeaderPaddingLeft	52	Gets or sets the left padding (space) of the column's header. *(Long expression)*
exHeaderPaddingRight	53	Gets or sets the right padding (space) of the column's header. *(Long expression)*
exHeaderPaddingTop	54	Gets or sets the top padding (space) of the column's header. *(Long expression)*
exHeaderPaddingBottom	55	Gets or sets the bottom padding (space) of the column's header. *(Long expression)*
exColumnResizeContiguously	64	Gets or sets a value that indicates whether the control's content is updated while the user is resizing the column. *(Boolean expression, False)*

constants DividerAlignmentEnum

Defines the alignment for a divider line into a divider item. Use the ItemDividerLineAlignment property to align the line in a divider item. Use the ItemDivider property to add a divider item.

Name	Value	Description
DividerBottom	0	The divider line is displayed on bottom side of the item.
DividerCenter	1	The divider line is displayed on center of the item.
DividerTop	2	The divider line is displayed at the top of the item.
DividerBoth	3	The divider line is displayed at the top and bottom of the item.

constants DividerLineEnum

Defines the type of divider line. The ItemDividerLine property uses the DividerLineEnum type. Use the ItemDivider property to define a divider item.

Name	Value	Description
EmptyLine	0	No line
SingleLine	1	Single line
DoubleLine	2	Double line
DotLine	3	Dotted line
DoubleDotLine	4	Double Dotted line
ThinLine	5	Thin line
DoubleThinLine	6	Double thin line

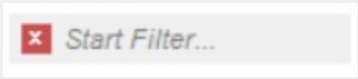
constants ExpandButtonEnum

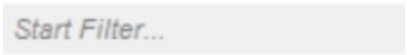
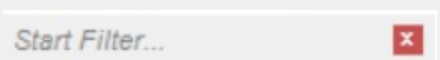
Defines how the control displays the expanding/collapsing buttons.

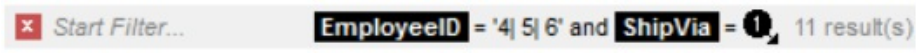
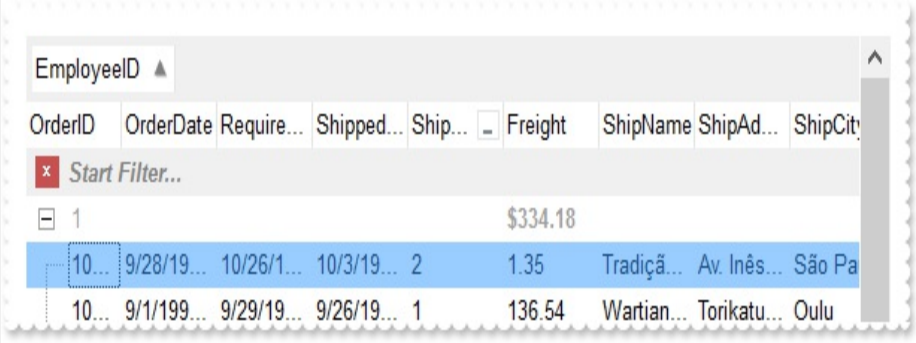
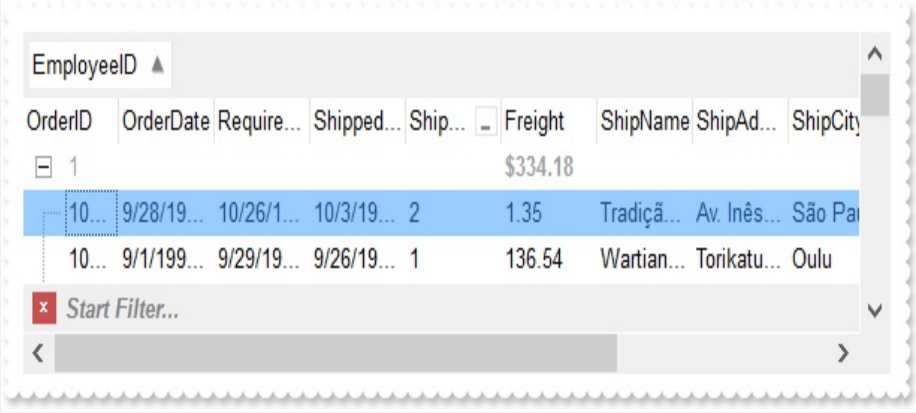
Name	Value	Description
exNoButtons	0	The control displays no expand buttons.
exPlus	-1	A plus sign is displayed for collapsed items, and a minus sign for expanded items.(⊞ ⊟)
exArrow	1	The control uses icons to display the expand buttons.(▸ ▾)
exCircle	2	The control uses icons to display the expand buttons. (⊕ ⊖)
exWPlus	3	The control uses icons to display the expand buttons. (✚ ▬)
exCustom	4	reserved

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. If the control's RightToLeft property is True, the close button of the control's filter bar would be automatically displayed on the left side. 

exFilterBarCompact	2048	The exFilterBarCompact flag compacts the control's filter bar, so the filter-prompt will be displayed to 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:

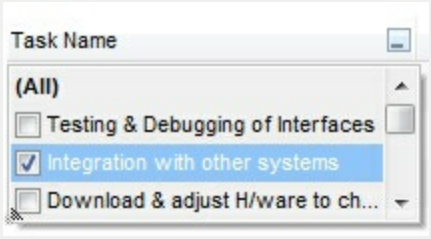
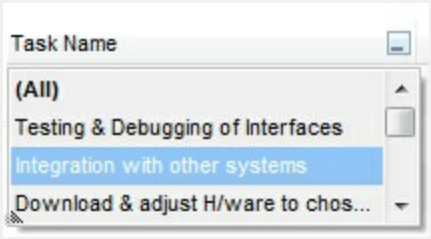
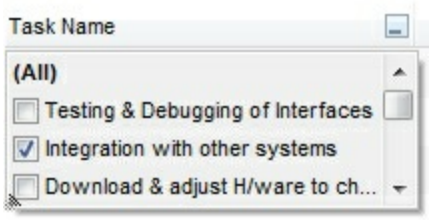
Name	Value	Description
exItemsWithoutChilds	0	Items (and parent-items) that match the filter are shown (no child-items are included)
exItemsWithChilds	1	Items (parent and child-items) that match the filter are shown
exRootsWithoutChilds	2	Only root-items (excludes child-items) that match the filter are 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.

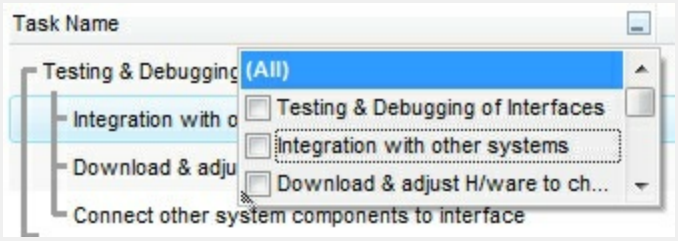
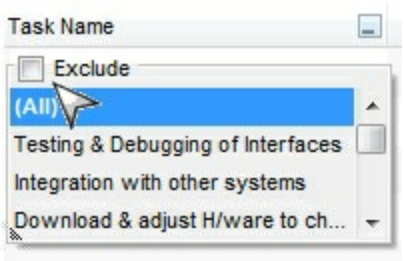
constants FilterListEnum

The FilterListEnum type specifies the type of items being included in the column's drop down list filter. The FilterList property specifies the items being included to the column's drop down filter-list, including other options for filtering. Use the DisplayFilterPattern and/or DisplayFilterDate property to display the pattern field, a date pattern or a calendar control inside the drop down filter window.

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. The SplitCell method adds an inner cell, on in other words splits a cell.
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.
exShowCheckBox	256	The filter's list displays a check box for each 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

exShowFocusItem	1024	item being focused in the drop down filter list is the one you have in the control's selection. This flag has 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

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 FilterTypeEnum

Defines the type of filter applies to a column. Use the FilterType property of the Column object to specify the type of filter being used. Use the Filter property of Column object to specify the filter being used. The value for Filter property depends on the FilterType property.

Name	Value	Description
exAll	0	No filter applied.
exBlanks	1	Only blank items are included.
exNonBlanks	2	Only non blanks items are included.
exPattern	3	Only items that match the pattern are included. The Filter property of the Column object defines the pattern. A pattern may contain the wild card characters '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character. The '|' character separates multiple patterns. If any of the *, ?, # or | characters are preceded by a \ (escape character) it masks the character itself.
exDate	4	Use the exDate type to filter items into a given interval. The Filter property of the Column object defines the interval of dates being used to filter items. The interval of dates should be as [dateFrom] to [dateTo]. If the dateFrom value is missing, the control includes only the items before the dateTo date, if the dateTo value is missing, the control includes the items after the dateFrom date. If both dates (dateFrom and dateTo) are present, the control includes the items between this interval of dates. For instance, the "2/13/2004 to" includes all items after 2/13/2004 inclusive, or "2/13/2004 to Feb 14 2005" includes all items between 2/13/2004 and 2/14/2004.
exNumeric	5	If the FilterType property is exNumeric, the Filter property may include operators like <, <=, =, <>, >= or > and numbers to define rules to include numbers in the control's list. For instance, the "> 10 < 100" filter indicates all numbers greater than 10 and less than 100. If the FilterType property is exNumeric, the drop down filter window doesn't

		display the filter list that includes items "(All)", "(Blanks)", ... and so on.
exCheck	6	Only checked or unchecked items are included. The CellState property indicates the state of the cell's checkbox. The control filters for checked items, if the Filter property is "1". The control filters for unchecked items, if the Filter property is "0". A checked item has the the CellState property different than zero. An unchecked item has the CellState property on zero.
exImage	10	Filters items by icons. The CellImage property indicates the cell's icon.
exFilter	240	Only the items that are in the Filter property are included. The CellCaption property indicates the cell's caption.
exFilterDoCaseSensitive	256	By default, the filtering is case-insensitive. If this flag is set, the filtering is case-sensitive. This option can be combined with exFilter or exPattern flag to perform a case-sensitive filtering. For instance, the exFilter + exFilterDoCaseSensitive indicates that the column includes only the values that match exactly the values in the Filter property.
exFilterExclude	512	exFilterExclude

constants FormatApplyToEnum

The FormatApplyToEnum expression indicates whether a format is applied to an item or to a column. Any value that's greater than 0 indicates that the conditional format is applied to the column with the value as index. A value less than zero indicates that the conditional format object is applied to items. Use the ApplyTo property to specify whether the conditional format is applied to items or to columns.

Name	Value	Description
exFormatToItems	-1	Specifies whether the condition is applied to items.
exFormatToColumns	0	Specifies whether the condition is applied to columns. The 0 value indicates that the conditional format is applied to the first column. The 1 value indicates the conditional format is applied to the second column. The 2 value indicates the conditional format is applied to the third column, and so on.

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 HitTestInfoEnum

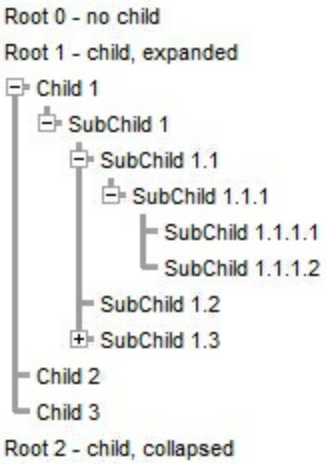
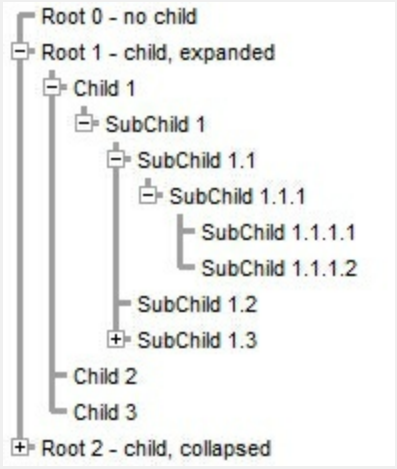
The HitTestInfoEnum expression defines the hit area within a cell. Use the ViewItemFromPoint property to determine the hit test code within the cell.

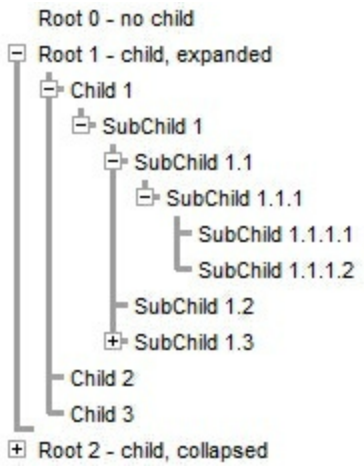
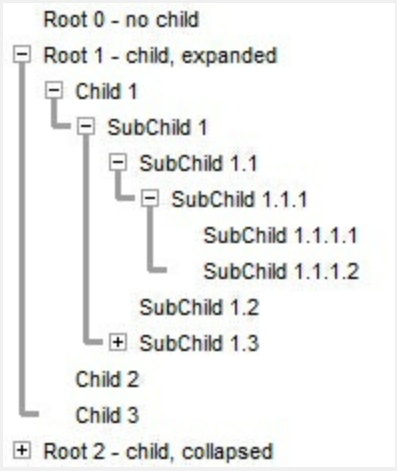
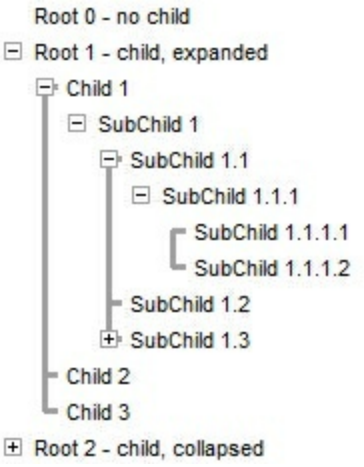
Name	Value	Description
exHTCell	0	In the cell's client area.
exHTExpandButton	1	In the +/- button associated with a cell. The HasButtons property specifies whether the cell displays a +/- sign to let user expands the item.
exHTCellIndent	2	In the indentation associated with a cell. The Indent property retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
exHTCellInside	4	On the icon, picture, check or caption associated with a cell.
exHTCellCaption	20	(HEXA 14) In the caption associated with a cell. The CellValue property specifies the cell's value.
exHTCellCheck	36	(HEXA 24) In the check/radio button associated with a cell. The CellHasCheckBox or CellHasRadioButton property specifies whether the cell displays a checkbox or a radio button.
exHTCellIcon	68	HEXA 44) In first icon associated with a cell. The CellImage or CellImages property specifies the cell's icon displayed next to the cell's caption.
exHTCellPicture	132	(HEXA 84). In a picture associated to a cell.
exHTCellCaptionIcon	1044	(HEXA 414) In the icon's area inside the cell's caption. The <img> tag inserts an icon inside the cell's caption. The <img> tag is valid only if the CellValueFormat property exHTML
exHTBottomHalf	2048	(HEXA 800) The cursor is in the bottom half of the row. If this flag is not set, the cursor is in the top half of the row. This is an OR combination with the rest of predefined values. For instance, you can check if the cursor is in the bottom half of the row using HitTestCode AND 0x800
exHTBetween	4096	The cursor is between two rows. This is an OR combination with the rest of predefined values. For instance, you can check if the cursor is between

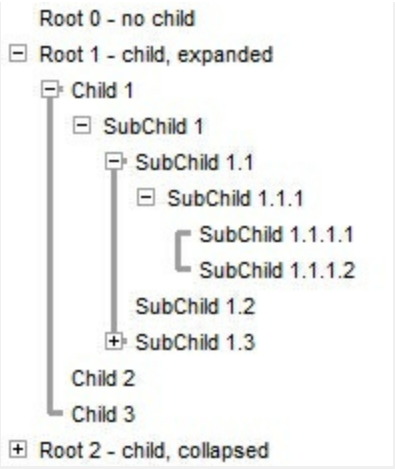
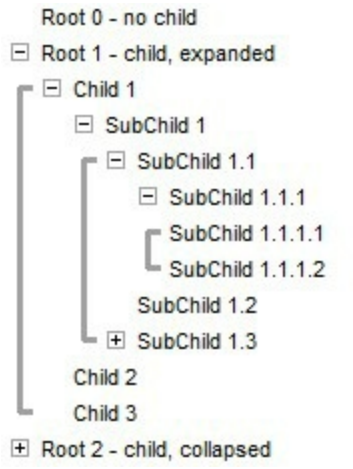
constants LinesAtRootEnum

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 InplaceAppearanceEnum

The InplaceAppearanceEnum type supports the following values:

Name	Value	Description
NoApp	0	No border
FlatApp	1	Flat
SunkenApp	2	Sunken
RaisedApp	3	Raised
EtchedApp	4	Etched
BumpApp	5	Bump
ShadowApp	6	Shadow
InsetApp	7	Inset
SingleApp	8	Single

constants NumericEnum

Use the Numeric property to specify the format of numbers when editing a field.

Name	Value	Description
exInteger	-1	Allows editing numbers of integer type. The format of the integer number is: **[+/-]digit**, where **digit** is any combination of digit characters. This flag can be combined with exDisablePlus, exDisableMinus or exDisableSigns flags. For instance, the 0x3FF (hexa representation, 1023 decimal) value indicates an integer value with no +/- signs.
exAllChars	0	Allows all characters. No filtering.
exFloat	1	Allows editing floating point numbers. The format of the floating point number is: **[+/-]digit[.digit[[e/E/d/D][+/-]digit]]**, where digit is any combination of **digit** characters. Use the exEditDecimalSymbol option to assign a new symbol for '**.**' character (decimal values). This flag can be combined with exDisablePlus, exDisableMinus or exDisableSigns flags.
exFloatInteger	2	Allows editing floating point numbers without exponent characters such as e/E/d/D, so the accepted format is **[+/-]digit[.digit]**. Use the exEditDecimalSymbol option to assign a new symbol for '**.**' character (decimal values). This flag can be combined with exDisablePlus, exDisableMinus or exDisableSigns flags.
exDisablePlus	256	Prevents using the + sign when editing numbers. If this flag is included, the user can not add any + sign in front of the number.
exDisableMinus	512	Prevents using the - sign when editing numbers. If this flag is included, the user can not add any - sign in front of the number.
exDisableSigns	768	Prevents using the +/- signs when editing numbers. If this flag is included, the user can not add any +/- sign in front of the number. For instance exFloatInteger + exDisableSigns allows editing floating points numbers without using the exponent and plus/minus characters, so the allowed format is

digit[.digit]

constants PictureDisplayEnum

Specifies how the picture is displayed on the control's background. Use the PictureDisplay property to specify how the control displays its picture.

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 ReadOnlyEnum

Use the Enabled property to disable the control.

Name	Value	Description
exReadWrite	0	(boolean False) The control allows changes. The user can use the cell's editor to change the cell's value.
exReadOnly	-1	(boolean True) The control is read only and the cell's editor is not visible.
exLocked	1	The control is read only, and the cell's editor is visible but locked. For instance, if the cell's editor contains a drop down portion, the user can display the drop down portion of the control, but it can't select a new value. Also, if the editor contains multiple buttons they are active as the control is not read only.

constants ScrollBarEnum

The ScrollBarEnum type specifies the vertical or horizontal scroll bar in the control. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bars

Name	Value	Description
exVScroll	0	Indicates the vertical scroll bar (cascade column view)
exHScroll	1	Indicates the horizontal scroll bar (cascade column view)
exScroll	2	Indicates the control's horizontal scroll bar.

constants ScrollBarsEnum

Specifies which scroll bars will be visible on a control. The ScrollBars property of the control specifies the scroll bars being visible in the control. By default, the ScrollBars property is exBoth, which indicates that both scroll bars of the component are being displayed only when they require.

- The horizontal scroll bar is not shown, if the ColumnAutoResize property is True, or if the ScrollBars property is exNoScroll. The horizontal scroll bar is shown if required, if the ScrollBars property is exBoth or exHorizontal, else it is always shown if the ScrollBars property is exDisableBoth or exDisableNoHorizontal
- The vertical scroll bar of the control is shown if required, if the ScrollBars is exBoth or exVertical, else if it is always shown if the ScrollBars property is exDisableBoth or exDisableVertical. For instance, if the ScrollBars property is exBoth OR exVScrollOnThumbRelease, the control's content is scrolled when the user releases the vertical thumb. If your data displays items with different heights, you should set the ScrollBySingleLine property on True.

Use the Scroll method to programmatically scroll the control's content to specified position. The ScrollPos property determines the position of the control's scroll bars. The ScrollWidth property specifies the width in pixels, of the vertical scroll bar. The ScrollHeight property specifies the height in pixels of the horizontal scroll bar. The ScrollOrderParts property specifies the order to display the parts of the scroll bar (buttons, thumbs and so on). The ScrollPartCaption property specifies the caption to be shown on any part of the scroll bar. Use the SelectPos property to select items giving its position.

The ScrollBars property supports a bitwise OR combination of the following values:

Name	Value	Description
exNoScroll	0	No scroll bars are shown. This flag should not be combined with any other.
exHorizontal	1	Only horizontal scroll bars are shown. This flag can be combined with any other flag greater or equal with 256.
exVertical	2	Only vertical scroll bars are shown. This flag can be combined with any other flag greater or equal with 256.
exBoth	3	Both horizontal and vertical scroll bars are shown. This flag can be combined with any other flag greater or equal with 256.
exDisableNoHorizontal	5	The horizontal scroll bar is always shown, it is disabled if it is unnecessary. This flag can be

		combined with any other flag greater or equal with 256.
exDisableNoVertical	10	The vertical scroll bar is always shown, it is disabled if it is unnecessary. This flag can be combined with any other flag greater or equal with 256.
exDisableBoth	15	Both horizontal and vertical scroll bars are always shown, disabled if they are unnecessary. This flag can be combined with any other flag greater or equal with 256.
exHScrollOnThumbRelease	256	Scrolls the control's content when the user releases the thumb of the horizontal scroll bar. Use this option to specify that the user scrolls the control's content when the thumb of the scroll box is released.
exVScrollOnThumbRelease	512	Scrolls the control's content when the user releases the thumb of the vertical scroll bar. Use this option to specify that the user scrolls the control's content when the thumb of the scroll box is released.
exHScrollEmptySpace	1024	Allows empty space, when control's content is horizontally scrolled to the end. If this flag is set, the last visible column, is displayed on leftmost position of the control, when the user horizontally scrolls to the end.
exVScrollEmptySpace	2048	Allows empty space, when control's content is vertically scrolled to the end. If this flag is set, the last visible item, is displayed on top of the control, when the user vertically scrolls to the end.

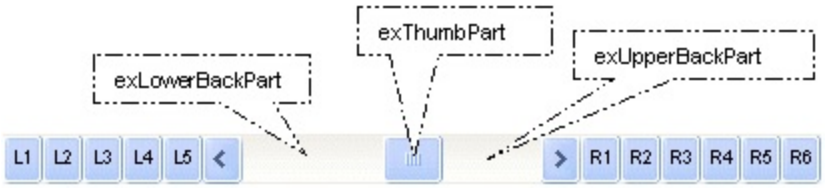
constants ScrollEnum

The ScrollEnum expression indicates the type of scroll that control supports. Use the Scroll method to scroll the control's content by code.

Name	Value	Description
exScrollUp	0	Scrolls up the control by a single line.
exScrollDown	1	Scrolls down the control by a single line.
exScrollVTo	2	Scrolls vertically the control to a specified position.
exScrollLeft	3	Scrolls the control to the left by a single pixel, or by a single column if the ContinueColumnScroll property is True.
exScrollRight	4	Scrolls the control to the right by a single pixel, or by a single column if the ContinueColumnScroll property is True.
exScrollHTo	5	Scrolls horizontaly the control to a specified position.

constants ScrollPartEnum

The ScrollPartEnum type defines the parts in the control's scrollbar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollPartCaption property to specify the caption being displayed in any part of the control's scrollbar.



Name	Value	Description
exExtentThumbPart	65536	The thumb-extension part.
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.
		(R1) The first additional button in the right or down

exRightB1Part	32	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
exRightB5Part	2	(R5) The fifth additional button in the right or down 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 SortOnClickEnum

Specifies the action that control takes when user clicks the column's header. The SortOnClick Property specifies whether the control sorts a column when its caption has been clicked.

Name	Value	Description
exNoSort	0	The column is not sorted when user clicks the column's header.
exDefaultSort	-1	The control sorts the column when user clicks the column's header.
exUserSort	1	The control displays the sort icons, but it doesn't sort the column.

constants SortOrderEnum

Specifies the column's order type. Use the SortOrder property to specify the column's sort order

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	The column gets sorted numerical using the CellData property.
SortCellData	6	The column gets sorted numerical using the CellSortData property.
SortCellDataString	7	The CellSortData property indicates the values being sorted. The values are sorted as string.
exSortByValue	16	The column gets sorted by cell's value rather than cell's caption.
exSortByState	32	The column gets sorted by cell's state rather than cell's caption.
exSortByImage	48	The column gets sorted by cell's image rather than cell's caption.

constants StatusBarAnchorEnum

The StatusBarAnchorEnum type specifies how the status bar is displayed relative to the control. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying. The StatusBarAnchorEnum type supports the following values:

Name	Value	Description
exStatusBarNone	0	The control's status bar is not visible.
exStatusBarAnchorBottom	1	The control's status bar is aligned to the bottom side of the control.
exStatusBarAnchorTop	2	The control's status bar is aligned to the top side of the control.
exStatusBarWordWrap	16	The status's label is displaying its content using word-wrap (multiple lines).

constants ItemsAllowSizingEnum

The ItemsAllowSizingEnum type specifies whether the user can resize items individuals or all items at once, at runtime. Use the ItemsAllowSizing property to specify whether the user can resize items individuals or all items at once, at runtime. Curently, the ItemsAllowSizingEnum type supports the following values:

Name	Value	Description
exNoSizing	0	The user can't resize the items at runtime.
exResizeItem	-1	Specifies whether the user resizes the item from the cursor.
exResizeAllItems	1	Specifies whether the user resizes all items at runtime.

constants UIVisualThemeEnum

The UIVisualThemeEnum expression specifies the UI parts that the control can shown using the current visual theme. The The UIVisualThemeEnum type supports following values:

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

constants VAlignmentEnum

Specifies the source's vertical alignment.

Name	Value	Description
exTop	0	exTop
exMiddle	1	exMiddle
exBottom	2	exBottom

constants ValueFormatEnum

Defines how the cell's value is shown. The CellValueFormat property indicates the way the cell displays its content. The Def(exCellValueFormat) property indicates the format for all cells within the column. The CellValue property indicates the cell's value, content or formula. The ComputedField property indicates the formula to compute all cells in the column. The FormatColumn property indicates the format to be applied for cells in the columns.The ValueFormatEnum type supports can be a combination of the following values:

Name	Value	Description
exText	0	Standard text. No HTML tags are displayed
		The control uses built-in HTML tags to display the caption using HTML format. The control 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

exHTML	1	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 $^{\text{subscript}}$ • **<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:

- **<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:

exComputedField	2	Indicates a computed field. The CellValue property indicates the formula to compute the field. A computed field can display its content using the values from any other cell in the same item/row. For instance %1 + %2 indicates that the cell displays the addition from the second and third cells in the same item (cells are 0 based). For instance, if the cells are of numeric format the result is the sum of two values, while if any of the cell is of string type it performs a concatenation of the specified cells. The ComputedField property indicates the formula to compute all cells in the column. The exComputedField can be combined with exText or exHTML. For instance, the exComputedField + exHTML indicates that the computed field may display HTML tags. The syntax for the CellValue property should be: **formula** where %n indicates the cell from the n-index. The operation being supported are listed bellow. For instance %1 + %2 indicates the sum of all cells in the second and third column from the current item.
		Indicates a total/subtotal field. The CellValue property indicates the formula for total field that includes an aggregate function such as: sum, min, max, count, avg. The exTotalField can be combined with exText or exHTML. For instance, the exTotalField + exHTML indicates that the total field may display HTML tags. The syntax for the CellValue property should be: **aggregate**(**list**,**direction**,**formula**) where: **aggregate** must be one of the following: • *sum* - calculates the sum of values. • *min* - retrieves the minimum value. • *max* - retrieves the maximum value. • *count* - counts the number of items.

- *avg* - calculates the average of values.

list must be one of the following:

- a *long* expression that specifies the index of the item being referred.
- a predefined string expression as follows:
  - *all* - indicates all items, so the formula is being applied to all items. The direction has no effect.
  - *current* - the current item.
  - *parent* - the parent item.
  - *root* - the root item.

direction must be one of the following:

- *dir* - collects the direct descendents.
- *rec* - collects the leaf descendents (leaf items). A leaf item is an item with no child items.
- *all* - collects all descendents.

Currently, the following items are excluded by aggregate functions:

exTotalField 4

- *not-sortable items*. The SortableItem property specifies whether the item can be sorted (a sortable item can change its position after sorting, while a not-sortable item keeps its position after sorting.
- *not-selectable items*. The SelectableItem property specifies whether the user can selects/focus the specified item.
- *divider items*. The ItemDivider property specifies whether the item displays a single cell, instead displaying whole cells.

In conclusion, aggregate functions counts ONLY items that are:

- *sortable*, SortableItem is True, by default.
- *selectable*, SelectableItem is True, by default.
- *not divider*, ItemDivider is -1, by default.

Shortly, by setting to a different value to any of

these properties, makes the item to be ignored by the aggregate functions.

For instance

- count(current,dir,1) counts the number of child items (not implies recursively child items).
- count(current,all,1) counts the number of all child items (implies recursively child items).
- count(current,rec,1) counts the number of leaf items (implies recursively leaf items).
- count(current,rec,1) counts the number of leaf items (a leaf item is an item with no child items).
- sum(parent,dir,%1=0?0:1) counts the not-zero values in the second column (%1)
- sum(parent,dir,%1 + %2) indicates the sum of all cells in the second (%1) and third (%2) column that are directly descendent from the parent item.
- sum(all,rec,%1 + %2) sums all leaf cells in the second (%1) and third (%2) columns.

The **formula** on the CellValue property (if the CellValueFormat property indicates the exComputedField or exTotalField) may include the formatting operators as follows:

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, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance,*

"%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.

- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

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

Usage examples:

1. "**1**", the cell displays 1
2. "**%0 + %1**", the cell displays the sum between cells in the first and second columns.
3. "**%0 + %1 - %2**", the cell displays the sum between cells in the first and second columns minus the third column.
4. "**(%0 + %1)*0.19**", the cell displays the sum between cells in the first and second columns multiplied with 0.19.
5. "**(%0 + %1 + %2)/3**", the cell displays the arithmetic average for the first three columns.
6. "**%0 + %1 < %2 + %3**", displays 1 if the sum between cells in the first two columns is less than the sum of third and forth columns.
7. "**proper(%0)**" formats the cells by capitalizing first letter in each word
8. "**currency(%1)**" displays the second column as currency using the format in the control panel for money
9. "**len(%0) ? currency(dbl(%0)) : ""** displays the currency only for not empty/blank cells.
10. "**int(date(%1)-date(%2)) + 'D ' + round(24*(date(%1)-date(%2) - floor(date(%1)-date(%2)))) + 'H'"** displays interval between two dates in days and hours, as xD yH
11. **"2:=((1:=int(0:= date(%1)-date(%0))) = 0 ? '' : str(=:1) + ' day(s)') + (3:=round(24*(=:0-floor(=:0))) ? (len(=:2) ? ' and ' : '') + =:3 + ' hour(s)' : '')"** displays the interval between two dates, as x day(s) [and y hour(s)], where the x indicates the number of days, and y the number of hours. The hour part is missing, if 0 hours is displayed, or nothing is displayed if dates are identical.

constants ViewItemStateEnum

The ViewItemStateEnum type specifies different states for an item. The ViewItemStateStartChanging / ViewItemStateEndChanging notifies your application that an item expanded or activated / selected, or when a check box has been clicked / changed. The ViewItemStateEnum type supports the following values:

Name	Value	Description
exExpandItem	1	An item is expanded or collapsed.
exCheckItem	2	An item is checked or unchecked.
exActivateItem	3	An item is activated / selected

constants ViewItemUpdateEnum

The ViewItemUpdate event notifies your application that a new item has been added or removed of the View object. The ViewItemUpdateEnum type supports the following values:

Name	Value	Description
exAddItem	1	The a new item has been added.
exAddGroupItem	2	Occurs after a new group Item has been inserted to Items collection.
exRemoveItem	3	An item is about to be removed.

constants ViewOperationEnum

The ViewOperationEnum type specifies operations that could start or end. The ViewStartChanging / ViewEndChanging events notify your application that an operation starts or ends. The ViewOperationEnum type supports the following values:

Name	Value	Description
exSplitViewChange	1	The user splits/resizes the view into multiple views.
exResizeCascadeColumn	2	The user resizes the cascade column.
exSelectionChange	3	The view selection is changing.
exDataSourceChange	4	The control's Data source is changing.
exLayoutChange	5	The view layout is changing.
exShowContextMenu	20	Occurs when the control is about to display the object's context menu.
exExecuteContextMenu	21	Occurs when the control is about to execute a command from the object's context menu.

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.
Skin as Variant	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.

- 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%.

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. 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.

On **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_CHECKE 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRESS CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
	BP_PUSHBUTTON = 1	PBS_NORMAL = 1 = 2 PBS_PRESSED PBS_DISABLED = PBS_DEFAULTED =
	BP_RADIOBUTTON = 2	RBS_UNCHECKED 1 RBS_UNCHECKE RBS_UNCHECKED = 3 RBS_UNCHECKED = 4 RBS_CHECKE 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
		EBSGE_NORMAL =

	EBP_SPECIALGROUPEXPAND = 11	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_PRESSE
	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
	MP_MENUDROPDOWN = 2	MS_NORMAL = 1 MS_SELECTED = 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
		MDS_NORMAL = 1 = 2 MDS_PRESSE

MENUBAND	MDP_NEWAPPBUTTON = 1	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	
SCROLLBAR	SBP_ARROWBTN = 1	ABS_DOWNDISAB ABS_DOWNHOT, ABS_DOWNNORM ABS_DOWNPRESS ABS_UPDISABLED ABS_UPHOT, ABS_UPNORMAL, ABS_UPPRESSED, ABS_LEFTDISABLE 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
	SBP_SIZEBOX = 10	SZB_RIGHTALIGN 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
		UPS_NORMAL = 1

	SPNP_UP = 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
	TABP_TABITEMBOTHEDGE = 4	TIBES_NORMAL = TIBES_HOT = 2 TIBES_SELECTED TIBES_DISABLED TIBES_FOCUSED =
	TABP_TABITEMLEFTEDGE = 2	TILES_NORMAL = TILES_HOT = 2 TILES_SELECTED TILES_DISABLED = TILES_FOCUSED =

	TABP_TABITEMRIGHTEDGE = 3	TIRES_NORMAL = TIRES_HOT = 2 TIRES_SELECTED TIRES_DISABLED TIRES_FOCUSED
	TABP_TOPTABITEM = 5	TTIS_NORMAL = 1 = 2 TTIS_SELECTE TTIS_DISABLED = TTIS_FOCUSED =
	TABP_TOPTABITEMBOTHEDGE = 8	TTIBES_NORMAL TTIBES_HOT = 2 TTIBES_SELECTE TTIBES_DISABLED TTIBES_FOCUSED
	TABP_TOPTABITEMLEFTEDGE = 6	TTILES_NORMAL = TTILES_HOT = 2 TTILES_SELECTED TTILES_DISABLED TTILES_FOCUSED
	TABP_TOPTABITEMRIGHTEDGE = 7	TTIRES_NORMAL TTIRES_HOT = 2 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
	TTP_CLOSE = 5	TTCS_NORMAL = 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 = 4 TUS_DISABLED = TUBS_NORMAL =

	TKP_THUMBBOTTOM = 4	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 CS_ACTIVE = 1 CS

WINDOW	WP_CAPTION = 1	= 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 = 3 HTS_DISABLED = 4
	WP_MAX_BUTTON	MAXBS_NORMAL = MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
	WP_MAXCAPTION = 5	MXCS_ACTIVE = 1 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 =
	MAXBS_NORMAL =

WP_SMALLMAXBUTTON	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 = 3 VTS_DISABLED = 4

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.

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.

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
```

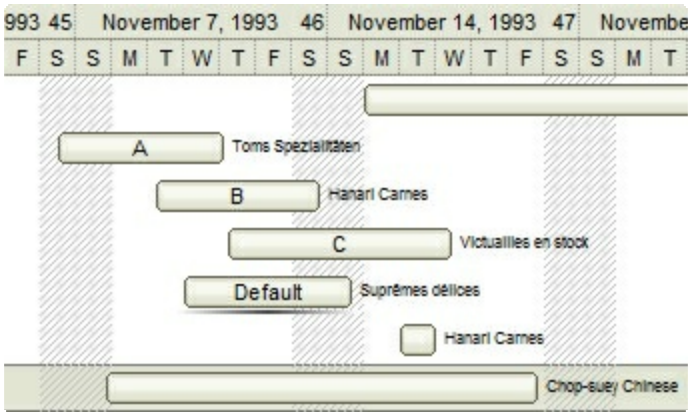
In the following screen shot the following objects displays the current EBN with a different color:

- "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

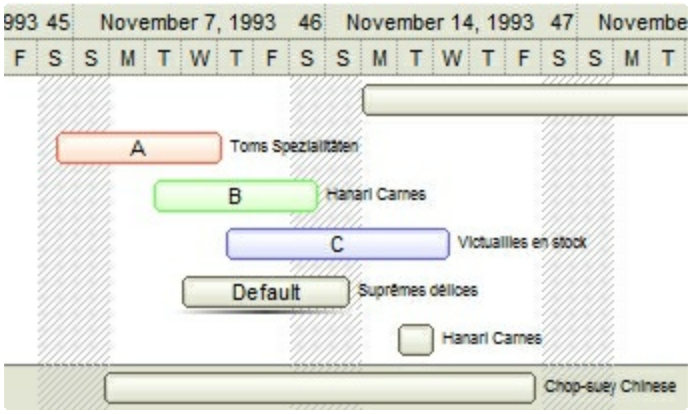
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

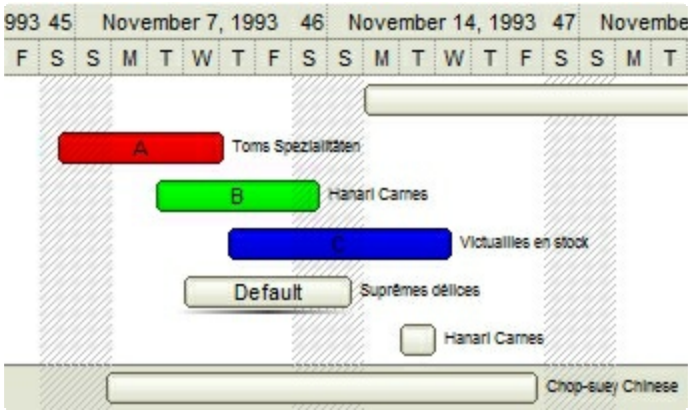
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



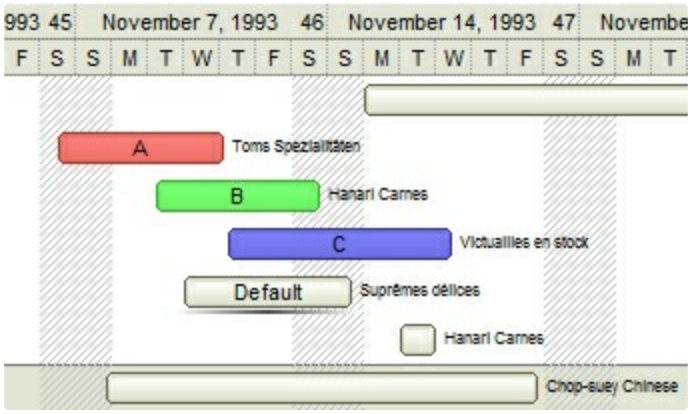
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the 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), ...)

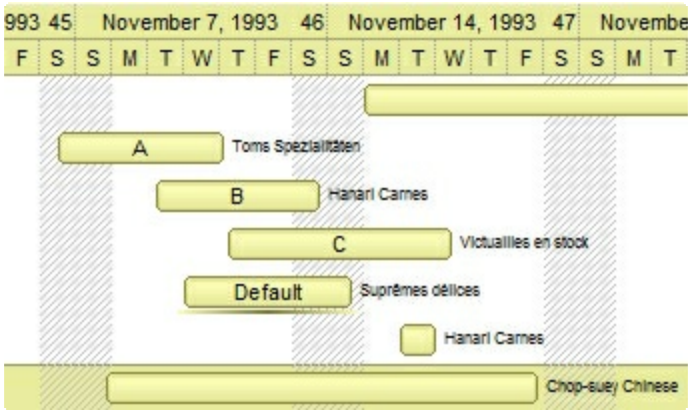


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

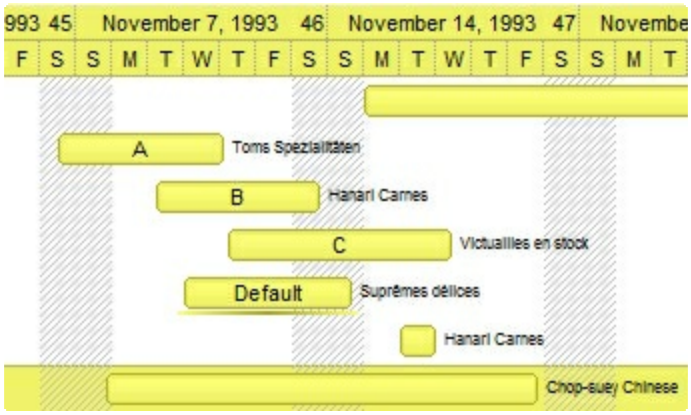


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on 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.

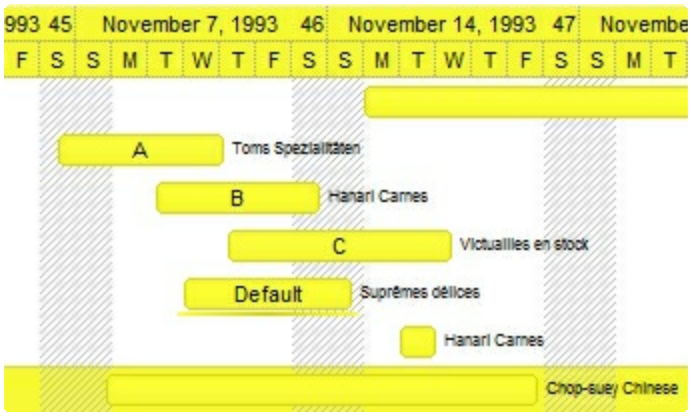
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



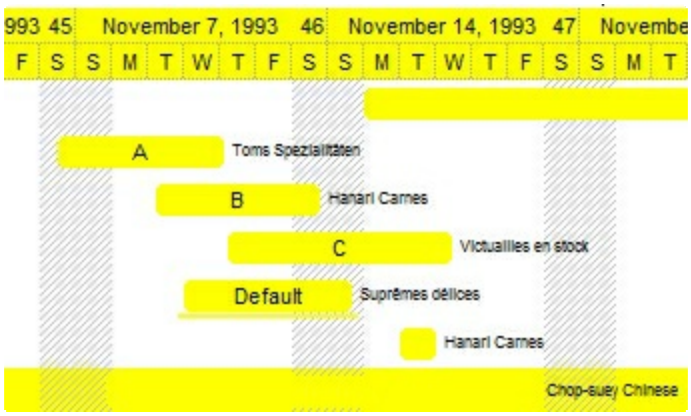
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



*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

CascadeTree 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: {4DD131BB-181C-428B-B0F3-8449ADA3AF49}. The object's program identifier is: "Exontrol.CascadeTree". The /COM object module is: "ExCascadeTree.dll"

The Miller columns (also known as Cascading Lists) are a browsing/visualization technique that can be applied to tree structures. The cascade columns allow multiple levels of the hierarchy to be open at once, and provide a visual representation of the current location. It is closely related to techniques used earlier in the Smalltalk browser, but was independently invented by Mark S. Miller in 1980 at Yale University. The CascadeTree object supports the following properties and methods:

Name	Description
AllowContextMenu	Enables or disables the file's context menu.
AllowSplitView	Specifies whether the user can split the control into multiple-views
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
BackColor	Specifies the control's background color.
BackColorAlternate	Specifies the control's alternate background color.
BackColorHeader	Specifies the header's background color.
BackColorLevelHeader	Specifies the multiple levels header's background color.
BackColorSortBar	Retrieves or sets a value that indicates the sort bar's background color.
BackColorSortBarCaption	Returns or sets a value that indicates the caption's background color in the control's sort bar.
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.

ColumnFromPoint	Retrieves the column from the point.
DataSource	Specifies the control's data as an array, XML, ADO or DAO.
DefaultView	Returns the control's default view.
DefColumnWidth	Specifies the width to create a new cascade column.
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.
ExecuteContextMenu	Executes a command from the object's context menu.
ExecuteTemplate	Executes a template and returns the result.
FilterBarBackColor	Specifies the background color of the control's filter bar.
FilterBarForeColor	Specifies the foreground color of the control's filter bar.
FitCascadeColumns	Retrieves or sets a value that indicates the number of cascading columns to fit.
FitToClient	Resizes or/and moves the all cascade columns to fit the control's client area.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
ForeColorAlternate	Specifies the control's alternate foreground color.
ForeColorHeader	Specifies the header's foreground color.
ForeColorSortBar	Retrieves or sets a value that indicates the sort bar'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.
FreezeEvents	Prevents the control to fire any event.
HeaderAppearance	Retrieves or sets a value that indicates the header's appearance.
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.
ItemFromPoint	Retrieves the item from the point.
Layout	Saves or loads the control's layout, such current selection for each panel, the widths of the cascade columns, and so on.
MaxColumnWidth	Specifies the maximum width for any cascade column.
MinColumnWidth	Specifies the minimum width for any cascade column.
Mode	Indicates whether the view allows single or multiple cascade columns.
Name	Selects the path using the name for each view.
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
Refresh	Refreses the control.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
ScrollButtonHeight	Specifies the height of the button in the vertical scrollbar.
ScrollButtonWidth	Specifies the width of the button in the horizontal scrollbar.
ScrollFont	Retrieves or sets the scrollbar's font.
ScrollHeight	Specifies the height of the horizontal scrollbar.
ScrollOrderParts	Specifies the order of the buttons in the scroll bar.
ScrollPartCaption	Specifies the caption being displayed on the specified scroll part.
ScrollPartEnable	Indicates whether the specified scroll part is enabled or disabled.
ScrollPartVisible	Indicates whether the specified scroll part is visible or hidden.
ScrollThumbSize	Specifies the size of the thumb in the scrollbar.
ScrollToolTip	Specifies the tooltip being shown when the user moves the scroll box.
ScrollWidth	Specifies the width of the vertical scrollbar.

SelBackColor	Retrieves or sets a value that indicates the selection background color.
Select	Selects the path using the key for each view.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowContextMenu	Specifies the object's context menu.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowToolTip	Shows the specified tooltip at given position.
SplitViewHeight	Specifies the height of split panels, separated by comma.
StatusBarHeight	Specifies the height of the control's status bar.
StatusBarLabel	Specifies the HTML label the control's status bar is displaying.
StatusBarVisible	Specifies whether the control's status bar is visible or hidden.
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.
UseTabKey	Retrieves or sets a value that specifies whether the Tab or SHIFT + Tab key navigates through the cascading columns.
Version	Retrieves the control's version.
View	Returns the view you are currently working on.
ViewColumnFromPoint	Retrieves the view and column from the point.
ViewFromPoint	Retrieves the view from the point.
ViewItemFromPoint	Retrieves the view and item from the point.

VisualAppearance	Retrieves the control's appearance.

property CascadeTree.AllowContextMenu as Boolean

Enables or disables the file's context menu.

Type	Description
Boolean	A boolean expression that indicates whether the control's context menu is enabled or disabled.

By default, the AllowContextMenu property is True. Use the AllowContextMenu to disable the control's context menu. The control's context menu is displayed when the user does a right click on the file or the folder. The system controls the items being inserted to the control's context menu. The ShowContextMenu property indicates the items to be displayed on the object's context menu. The ShowContextMenu property can be used to disable, update, remove or add new items. The ExecuteContextMenu property specifies the identifier of the command to be executed (id option in the ShowContextMenu property).

property CascadeTree.AllowSplitView as AllowSplitViewEnum

Specifies whether the user can split the control into multiple-views

Type	Description
AllowSplitViewEnum	An AllowSplitViewEnum expression whether the control supports multiple views (arranged vertically)

By default, the AllowSplitView property is exNoSplitView, so no additional view is supported. The AllowSplitView property specifies whether the user can split the control into multiple-views. The SplitViewHeight property specifies the height of split panels, separated by comma. The Background(exHSplitBar) property specifies the visual appearance of the control's split bar (horizontal split bar)

property CascadeTree.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.

property CascadeTree.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 files/folders, scrollbars are always shown in the control's client area. The skin may contain transparent objects, and so you can define round corners. Use the eXButton's Skin builder to view or change this file*

Use the Appearance property to specify the control's border. The Add method to add new skins to the control. Use the BackColor property to specify the control's background color. The BorderWidth / BorderHeight property specifies the size of the control's border.

method CascadeTree.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 CascadeTree1_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 CascadeTree.BackColor as Color

Retrieves or sets the control's background.

Type	Description
Color	A color expression that indicates the control's background color.

Use the BackColor / BackColorAlternate property to specify the control's background color. The Background property returns or sets a value that indicates the background color for parts in the control. Use the ForeColor property to change the control's foreground color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items.

property CascadeTree.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. If the first byte of four is 7F, the color is applied to the items section only. For instance, a value of 0x7F0000FF indicates that the BackColorAlternate property is red, and it applied to the items section only, so the non-items section is not painted.

By default, the control's BackColorAlternate property is zero. Use the BackColorAlternate property to specify the background color used to display alternate items in the control. 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. Use the ItemBackColor property to specify the item's background color. Use the CellBackColor property to specify the cell's background color. Use the Def(exCellBackColor) property to specify the background color for all cells in the column. If the first two bytes of the BackColorAlternate property are 0x7F, the non-items area is not filled.

property CascadeTree.BackColorHeader as Color

Specifies the header's background color.

Type	Description
Color	A color expression that indicates the background color of the control's header bar. The last 7 bits in the high significant byte of the color 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 BackColorHeader and ForeColorHeader properties to define colors used to paint the control's header bar. Use the HeaderVisible property to show or hide the control's header. Use the HeaderHeight property to specify the height of the control's header bar. Use the LevelKey property to allow multiple levels header bar. Use the BackColorLevelHeader property to specify the background color of the header when it displays multiple levels. Use the HeaderHeight property to specify the height of the header bar.

property CascadeTree.BackColorLevelHeader as Color

Specifies the multiple levels header's background color.

Type	Description
Color	A color expression that indicates the background color of the control's header bar.

Use the BackColorHeader and ForeColorHeader properties to define colors used to paint the control's header bar. Use the BackColorLevelHeader property to specify the background color of the control's header bar when multiple levels are displayed. Use the LevelKey property to display the control's header bar using multiple levels. If the control displays the header bar using multiple levels the HeaderHeight property gets the height in pixels of a single level in the header bar. The control's header displays multiple levels if there are two or more neighbor columns with the same non empty level key.

property CascadeTree.BackColorSortBar as Color

Retrieves or sets a value that indicates the sort bar's background color.

Type	Description
Color	A color expression that indicates the background color of the sort bar.

Use the BackColorSortBar property to specify the background color of the control's sort bar. Use the SortBarVisible property to show the control's sort bar. Use the BackColorSortBarCaption property to specify the background color of the caption of the sort bar. The caption of the sort bar is visible, if there are no columns in the sort bar. Use the SortBarCaption property to specify the caption of the sort bar. Use the ForeColorSortBar property to specify the foreground color of the control's sort bar. Use the BackColor property to specify the control's background color. Use the BackColorHeader property to specify the background color of the control's header bar. Use the BackColorLevelHeader property to specify the background color of the control's header bar when multiple levels are displayed.

property CascadeTree.BackColorSortBarCaption as Color

Returns or sets a value that indicates the caption's background color in the control's sort bar.

Type	Description
Color	A color expression that indicates the caption's background color in the control's sort bar.

Use the SortBarCaption property to specify the caption of the sort bar, when the control's sort bar contains no columns. Use the BackColorSortBar property to specify the background color of the control's sort bar. Use the ForeColorSortBar property to specify the foreground color of the caption in the control's sort bar.

property CascadeTree.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 Refresh method to refresh the control.

method CascadeTree.BeginUpdate ()

Prevents the control from painting until the EndUpdate method is called.

Type	Description

The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Use BeginUpdate and EndUpdate statement each time when the control requires more changes. Using the BeginUpdate and EndUpdate methods increase the speed of changing the control properties by preventing it from painting during changing.

property CascadeTree.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 control's border.

The BorderWidth / BorderHeight property specifies the size of the control's border. The Appearance property retrieves or sets the control's appearance. The Add method to add new skins to the control. Use the BackColor property to specify the control's background color.

property CascadeTree.BorderWidth as Long

Sets or retrieves a value that indicates the border width of the control.

Type	Description
Long	A Long expression that indicates the border width of the control.

By default, the BorderWidth property is 0. The BorderWidth / BorderHeight property specifies the size of the control's border. The Appearance property retrieves or sets the control's appearance. The Add method to add new skins to the control. Use the BackColor property to specify the control's background color.

property CascadeTree.ColumnFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

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 Y location of the mouse pointer. The y values is always expressed in client coordinates.
Long	A long expression that specifies the index of the column from the point.

The ColumnFromPoint property retrieves the column from the point. The ColumnFromPoint(-1,-1) property retrieves the column from the current cursor position. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point.

property CascadeTree.DataSource as Variant

Specifies the control's data as an array, XML, ADO or DAO.

Type	Description
Variant	A VARIANT expression that could be a string, an object as explained bellow.

The control can automatically handle Array, XML, ADO, DAO, DataSet through the DataSource properties (control and view objects). You can specify the data source for the entire control through the DataSource property, or for a particular view using View.DataSource property. If an internal error occurs while using the DataSource property the Error event occurs.

For instance,

- "...\sample.xml" opens the sample.xml file
- "...\sample.dbf" opens the specified sample.dbf table
- "Data Member=SELECT * FROM Orders ; Data Source=...\sample.accdb", opens the Orders table of the specified sample.accdb database
- "Data Member=SELECT * FROM Orders ; Data Source=...\sample.mdb", opens the Orders table of the specified sample.mdb database
- "Data Member=Orders ; Driver={Microsoft Access Driver (*.mdb)} ; DBQ=...\sample.mdb", opens the Orders table of sample.mdb database, using ODBC
- "Data Member=Orders ; Driver={Microsoft Access Driver (*.mdb)} ; DBQ=...\sample.mdb", opens the Orders table of sample.mdb database, using ODBC
- "Data Member=SELECT * FROM [Sheet1$] ; Driver={Microsoft Excel Driver (*.xls)} ; DBQ=...\sample.xls ; DriverID=790" reads the Sheet1 worksheet of the sample.xml file (Excel)
- "Source=...\sample.mdb;Member=Select * FROM Countries;Key=CountryCode;Tag=Country;Name=CountryName >>> Member=Select * FROM States WHERE CountryCode IN (<%Parent.CountryCode%>);Key=StateCode;Name=StateName;Tag=State ||| Member=Select * FROM Cities WHERE CountryCode IN (<%Parent.CountryCode%>);Tag=City;Name=Name >>> Member=Select * FROM Cities WHERE CountryCode IN (<%Parent.Parent.CountryCode%>) AND StateCode IN (<%Parent.StateCode%>);Tag=City;Name=Name", specifies multiple-data sources for Country, State and City views

where ... indicates the full path to the sample file.

The control's DataSource property in BNF syntax is:

```
<DataSource> ::= <DataSourceView> [ ">>>" <DataSourceView> ]
```

```
<DataSourceView> ::= <AltDataSourceView> [ "|||" <AltDataSourceView> ]
<AltDataSourceView> ::= <DataField> [";" <DataField>]
<DataField> ::= <DataFieldName> "=" <DataFieldValue>
<DataFieldName> ::= ["Data "]"Source" | ["Data "]"Member" | ["Data "]"Key" | ["Data "]"Tag" | ["Data "]"Name" | <ExtraDataFieldName>
<ExtraDataFieldName> ::= any extra field
<DataFieldValue> ::= field's value
```

In other words, the DataSource property provides data source for each view separated by >>> sequence, and for each view different alternatives to create the view separated by ||| sequence. The DataSource can include sequences between **<%** and **%>** which are filled at runtime, based on the current selection in all views.

Let's examine the following DataSource sequence:

> "**Source**=...\cities.mdb ; **Member**=Select * FROM Countries ; **Key**=CountryCode ; **Tag**=Country ; **Name**=CountryName >>> **Member**=Select * FROM States WHERE CountryCode IN (*<%Parent.CountryCode%>*) ; **Key**=StateCode ; **Name**=StateName ; **Tag**=State ||| **Member**=Select * FROM Cities WHERE CountryCode IN (*<%Parent.CountryCode%>*) ; **Tag**=City ; **Name**=Name >>> **Member**=Select * FROM Cities WHERE CountryCode IN (*<%Parent.Parent.CountryCode%>*) AND StateCode IN (*<%Parent.StateCode%>*) ; **Tag**=City ; **Name**=Name "

which can generate data source for up to 3 views red (country), green (state/city) and blue(city) as follows:

> "**Source**=...\cities.mdb ; **Member**=Select * FROM Countries ; **Key**=CountryCode ; **Tag**=Country ; **Name**=CountryName >>> **Member**=Select * FROM States WHERE CountryCode IN (*<%Parent.CountryCode%>*) ; **Key**=StateCode ; **Name**=StateName ; **Tag**=State ||| **Member**=Select * FROM Cities WHERE CountryCode IN (*<%Parent.CountryCode%>*) ; **Tag**=City ; **Name**=Name >>> **Member**=Select * FROM Cities WHERE CountryCode IN (*<%Parent.Parent.CountryCode%>*) AND StateCode IN (*<%Parent.StateCode%>*) ; **Tag**=City ; **Name**=Name "

At runtime, these three views may shows as:

Country View	Code
Turkey	TR
Turkmenistan	TM
Turks and Caicos Islands	TC
Tuvalu	TV
Uganda	UG
Ukraine	UA
United Arab Emirates	AE
United Kingdom	GB
United States	US
United States Minor Outlying...	UM
Uruguay	UY
Uzbekistan	UZ
Vanuatu	VU
Venezuela	VE
Viet Nam	VN
Virgin Islands, British	VG
Virgin Islands, U.S.	VI
Wallis and Futuna	WF
Western Sahara	EH

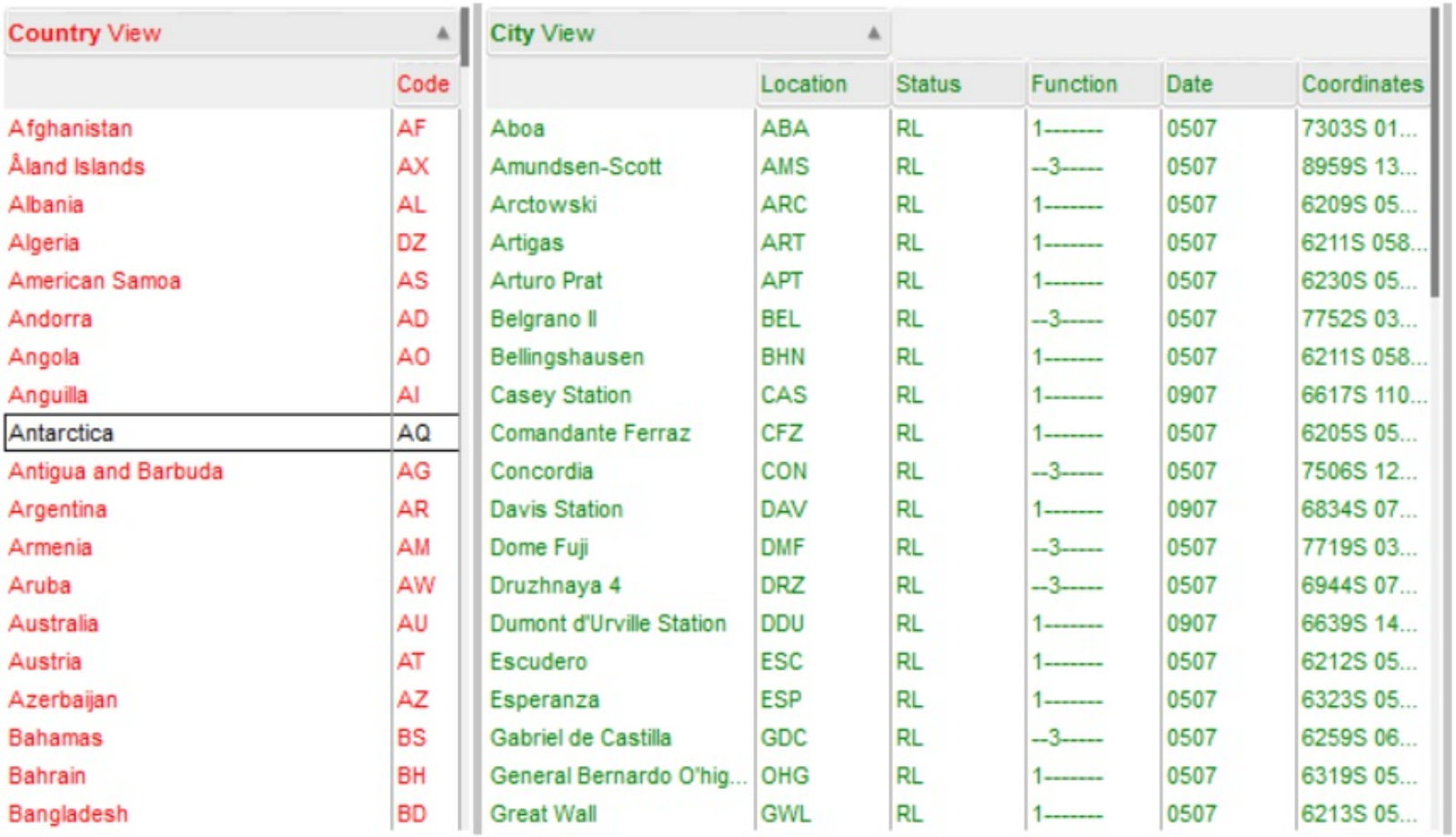
State View	Code
Alabama	AL
Alaska	AK
American Samoa (see also ...	AS
Arizona	AZ
Arkansas	AR
California	CA
Colorado	CO
Connecticut	CT
Delaware	DE
District of Columbia	DC
Florida	FL
Georgia	GA
Guam (see also separate e...	GU
Hawaii	HI
Idaho	ID
Illinois	IL
Indiana	IN
Iowa	IA
Kansas	KS

City View	Location	Status	Function	Date	Coordinates
Acton	T5C	RL	--3-----	1607	3939N 08558W
Adams City	YP2	RL	--3-----	1407	3949N 10455W
Advance	A2D	RL	--3-----	0901	3959N 08637W
Aiea (Oahu)	AIE	RQ	--3-----	9307	
Ajo	AJ9	RL	--3-----	1307	3222N 11251W
Akron	AKC	RL	--3-----	0212	4102N 08602W
Akron	AKO	AI	---4----	0001	
Alachua	AHU	RL	1-3-----	0212	2947N 08229W
Alamosa	ALS	AI	---4----	0001	
Albany	AB5	RQ	--3--6--	1001	4018N 08515W
Albion	YAB	RL	--3-----	0212	4124N 08525W
Alexandria	AXD	RQ	--3-----	9307	
Alma	AM2	RL	--3-----	1207	3917N 10603W
Altamonte Springs	ASP	RQ	--3-----	9307	
Altha	A7H	RL	--3-----	1501	3034N 08507W
Altoona	AZQ	RQ	--3-----	9307	
Alva	ALF	RL	--3-----	0212	2642N 08136W
Alys Beach	A5S	RL	--3-----	1701	3017N 08601W
Amado	AZ6	RL	--3-----	1407	3142N 11103W

In the same time, each view's DataSource shows as:

- View("Country").DataSource = "**Member**=Select * FROM Countries ; **Key**=CountryCode ; **Tag**=Country ; **Name**=CountryName ; **Source**=...\cities.mdb"
- View("State").DataSource = "**Member**=Select * FROM States WHERE CountryCode IN (*'US'*) ; **Key**=StateCode ; **Name**=StateName;**Tag**=State ; **Source**=...\cities.mdb"
- View("City").DataSource = "**Member**=Select * FROM Cities WHERE CountryCode IN (*'US'*) AND StateCode IN (*'AZ','FL','CO','HI','IN'*) ; **Tag**=City ; **Name**=Name ; **Source**=...\cities.mdb"

For instance, Antartica has no states, the **Member**=Select * FROM States WHERE CountryCode IN (*<%Parent.CountryCode%>*) generates no results, and so the alternative data source is used **Member**=Select * FROM Cities WHERE CountryCode IN (*<%Parent.CountryCode%>*), so at runtime the views may shows as:



Country View	Code
Afghanistan	AF
Åland Islands	AX
Albania	AL
Algeria	DZ
American Samoa	AS
Andorra	AD
Angola	AO
Anguilla	AI
Antarctica	AQ
Antigua and Barbuda	AG
Argentina	AR
Armenia	AM
Aruba	AW
Australia	AU
Austria	AT
Azerbaijan	AZ
Bahamas	BS
Bahrain	BH
Bangladesh	BD

City View	Location	Status	Function	Date	Coordinates
Aboa	ABA	RL	1-------	0507	7303S 01...
Amundsen-Scott	AMS	RL	--3-----	0507	8959S 13...
Arctowski	ARC	RL	1-------	0507	6209S 05...
Artigas	ART	RL	1-------	0507	6211S 058...
Arturo Prat	APT	RL	1-------	0507	6230S 05...
Belgrano II	BEL	RL	--3-----	0507	7752S 03...
Bellingshausen	BHN	RL	1-------	0507	6211S 058...
Casey Station	CAS	RL	1-------	0907	6617S 110...
Comandante Ferraz	CFZ	RL	1-------	0507	6205S 05...
Concordia	CON	RL	--3-----	0507	7506S 12...
Davis Station	DAV	RL	1-------	0907	6834S 07...
Dome Fuji	DMF	RL	--3-----	0507	7719S 03...
Druzhnaya 4	DRZ	RL	--3-----	0507	6944S 07...
Dumont d'Urville Station	DDU	RL	1-------	0907	6639S 14...
Escudero	ESC	RL	1-------	0507	6212S 05...
Esperanza	ESP	RL	1-------	0507	6323S 05...
Gabriel de Castilla	GDC	RL	--3-----	0507	6259S 06...
General Bernardo O'hig...	OHG	RL	1-------	0507	6319S 05...
Great Wall	GWL	RL	1-------	0507	6213S 05...

In the same time, each view's DataSource shows as:

- View("Country").DataSource = "**Member**=Select * FROM Countries ; **Key**=CountryCode ; **Tag**=Country ; **Name**=CountryName ; Source=...\cities.mdb"
- View("City").DataSource = "**Member**=Select * FROM Cities WHERE CountryCode IN (*'AQ'*) ; **Tag**=City ; **Name**=Name ; Source=...\cities.mdb"

Internally, the control's DataSource builds the view's DataSource with code as follows:

```
Private Sub CascadeTree1_CreateView(ByVal View As Object)
   With View
      Select Case View.Index
         Case 1: ' State or City
            .DataSource = CurrentDb.OpenRecordset("Select * FROM States WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
            .Key = "StateCode"
            .Name = "StateName"
            .Tag = "State"
            If (.Items.ItemCount = 0) Then
               .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
```

```
            .Key = ""
            .Tag = "City"
            .Name = "Name"
          End If
        Case 2: ' City
          .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ParentView.ValueList("CountryCode") & ") AND
StateCode IN (" & .ParentView.ValueList("StateCode") & ")")
          .Key = ""
          .Tag = "City"
          .Name = "Name"
      End Select
   End With
End Sub

Private Sub Form_Load()
   With CascadeTree1
      .BeginUpdate
      With .DefaultView
         .DataSource = CurrentDb.OpenRecordset("SELECT * FROM Countries")
         .Key = "CountryCode"
         .Tag = "Country"
         .Name = "CountryName"
      End With
      .EndUpdate
   End With
End Sub
```

The DataSource property supports the following fields:

- **Source** or **Data Source**, specifies the data source type, usually a MDB or ACCDB, but could be XLS, TXT or else. For instance, "**Source**=C:\Program Files\Exontrol\ExCascadeTree\Sample\Access\cities.accdb" refers to the cities.accdb database. The Source or Data Source field is required, else an error occurs (see Error event).
- **Member** of **Data Member**, indicates the SELECT SQL statement to be created from the Data Source. For instance, "**Member**=Select * FROM Countries", creates a view with all records from the Countries table. The Member or Data Member field is required, else an error occurs (see Error event). While this is not a requirement the

DataSource can include ANYWHERE sequences between **<%** and **%>** characters that specifies a runtime-generated string based on the current selection into your views. For instance, "**Member**=Select * FROM Cities WHERE CountryCode IN (**<%**Parent.Parent.CountryCode**%>**) AND StateCode IN (**<%**Parent.StateCode**%>**)"

Each sequence between **<%** and **%>** characters indicates the value of the current selection into a specified View of the specified Column/Field, and must be of the following BNF syntax:

```
<value> ::= "<%" <view> "." <column> "%>"
<view> ::= <parentview> [ "." <parentview> | <indexview>
<parentview> := "Parent"
<indexview> := 0 | 1 | 2 | ...
<column> := <index> | <name>
<index> := any index of any field into the Data Member
<name> := any name of the field into the Data Member / any caption of the column into the view
```

For instance:

- **<%**Parent.StateCode**%>**, indicates the value of the selection into the parent view of the column "StateCode".
- **<%**0.CountryCode**%>**, indicates the value of the selection into the view with the index 0 of the column "CountryCode".
- **<%**Parent.Parent.CountryCode**%>**, indicates the value of the selection into the parent of the parent view of the column "CountryCode".
- **<%**1.2**%>**, indicates the value of the selection into view with the index 1, on the column with the index 2

The View.ValueList property generates the values on the specified column (Key column) for all selected items, separated by , (comma) character. The ValueList property automatically includes ' character for strings and # for date fields. For instance, "'AZ','FL','CO','HI','IN'" or "1,2"

- **Key** or **Data Key**, specifies the index or the name of the field from the Data Member that generates keys for the current view. For instance, "**Key**=CountryCode" specifies that the CountryCode field of the current view generates keys for next child views. The Key property of the View object can be used to access the key of the view at runtime. If the Key refers to an exiting field/column in the current view, it means that the control generates the next view, once the user selects one or more items into the current view. If the Key is empty or points to an non-existing field/column in the current view, no view will be generated once an item in the current view is selected. The control fires the

CreateView event once a new view requires to be created. The ViewStartChanging(exSelectChange) / ViewEndChanging(exSelectChange) event notifies your application once the selection into the view is changing. During any event, you can access the view that generated the event, using the View property of the control. The Select property of the control generates the path of the current selection for all views using the Key property of each View (separated by \ backslash character). For instance, the Select property could return "US\AK". The Key field is not required, and if missing no view will be generated once the user selects an item into the current view.

- **Name** or **Data Name**, indicates the index or the name of the field from the Data Member, that generates names for the Name property. For instance, "**Name**=CountryName", indicates that the CountryName column of the current view generate values for the Name property. The Name property of the control generates the path of the current selection for all views using the Name property of each View (separated by \ backslash character). For instance, the Name property could return "United States\Alaska\Anchorage", The Name field is not required. By default, the column with the index 0 specifies the name column.
- **Tag** or **Data Tag**, specifies any extra data associated with the view. For instance, "**Tag**=Country". The Tag property of the View can be used to access the tag of the view at runtime. The Tag field is not required.
- or any additional field like **DBQ**, **Driver**, **DriverID, Server, SourceType, SourceDB** as specified by your connection string.

The Error event notifies your application once any internal error occurs. You can use the Description parameter of the Error event to find out more information about connection your data to the control.

Also the DataSource supports any of the following type of objects:

Safe-Array:

```
With CascadeTree1.DefaultView
   .LinesAtRoot = exLinesAtRoot
   .DataSource = Array("Item 1", Array("Sub-Item 1", "Sub-Item 2"), "Item 2")
End With
```

/NET or /WPF Version (VB)

```
With Excascadetree1.DefaultView
   .LinesAtRoot = exLinesAtRoot
   .DataSource = New Object() {"Item 1", New Object() {"Sub-Item 1", "Sub-Item 2"},
"Item 2"}
```

```
End With
```

/NET or /WPF Version (C#)

```
object[] items = {"Item 1", new object[] {"Sub-Item 1", "Sub-Item 2"}, "Item 2"};
excascadetree1.DataSource = items;
```

adds items from a safe-array. If it includes inside-safe arrays, it adds child items, and so on.

XML file name, a URL, an IStream, an IXMLDOMDocument

```
With CascadeTree1
   .DataSource = "C:\Program Files\Exontrol\ExCascadeTree\Sample\testing.xml"
End With
```

or:

```
With CascadeTree1.DefaultView
   .DataSource = "C:\Program Files\Exontrol\ExCascadeTree\Sample\testing.xml"
End With
```

or:

```
With CascadeTree1.DefaultView
   Set xml = CreateObject("MSXML.DOMDocument")
   With xml
      .Load "C:\Program Files\Exontrol\ExCascadeTree\Sample\testing.xml"
   End With
   .DataSource = xml
End With
```

ADO (Jet):

```
With CascadeTree1
   Set ado = CreateObject("ADODB.Recordset")
   With ado
      .Open "Countries", "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\Exontrol\ExCascadeTree\Sample\Access\cities.mdb", 3, 3
   End With
```

```
   .DataSource = ado
End With
```

or:

```
With CascadeTree1
   Set ado = CreateObject("ADODB.Recordset")
   With ado
      .Open "Countries", "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\Exontrol\ExCascadeTree\Sample\Access\cities.mdb", 3, 3
   End With
   .DefaultView.DataSource = ado
End With
```

ADO (OLEDB):

```
With CascadeTree1
   Set ado = CreateObject("ADODB.Recordset")
   With ado
      .Open "Countries", "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Program Files\Exontrol\ExCascadeTree\Sample\Access\cities.accdb", 3, 3
   End With
   .DataSource = ado
End With
```

or:

```
With CascadeTree1
   Set ado = CreateObject("ADODB.Recordset")
   With ado
      .Open "Countries", "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Program Files\Exontrol\ExCascadeTree\Sample\Access\cities.accdb", 3, 3
   End With
   .DefaultView.DataSource = ado
End With
```

DAO:

```
With CascadeTree1
```

```
  .DataSource = CurrentDb.OpenRecordset("Countries")
End With
```

or:

```
With CascadeTree1.DefaultView
  .DataSource = CurrentDb.OpenRecordset("Countries")
End With
```

As Microsoft Access uses DAO, you need to use the View's DataSource property rather than control's DataSource property as in the following sample:

```
Private Sub CascadeTree1_CreateView(ByVal View As Object)
  With View
    Select Case .Index
      Case 1: ' State or City
        .DataSource = CurrentDb.OpenRecordset("Select * FROM States WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
        .Tag = "State"
        .Key = "StateCode"
        .Name = "StateName"
        If (.Items.ItemCount = 0) Then
          .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
          .Tag = "City"
          .Key = ""
          .Name = "Name"
          .ColumnAutoResize = False
        End If
      Case 2: ' City
        .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ParentView.ValueList("CountryCode") & ") AND
StateCode IN (" & .ParentView.ValueList("StateCode") & ")")
        .Tag = "City"
        .Key = ""
        .Name = "Name"
    End Select
  End With
```

```
End Sub

Private Sub Form_Load()
   With CascadeTree1.DefaultView
   .DataSource = CurrentDb.OpenRecordset("SELECT * FROM Countries")
   .Tag = "Country"
   .Key = "CountryCode"
   .Name = "CountryName"
   End With
End Sub
```

The sample loads the Countries table into the default view (view with the index 0). Once the user clicks / selects / activates an item, the control creates a new view (with the index 1, 2 and so on) and fires the CreateView event. During the CreateView event you can load data from different tables based on the parent's view selection. See the ParentView.ValueList

property CascadeTree.DefaultView ([Level as Variant]) as View

Returns the control's default view.

Type	Description
Level as Variant	A long expression that specifies the vertical level to access its default view. If missing, the default view for the first level is returned (0-based). The AllowSplitView property specifies whether the user can split the control into multiple-views. The SplitViewHeight property specifies the height of split panels, separated by comma.
View	A View object that specifies the default view.

The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs. The ActiveView property gets the active view (the last view with any active items inside). The CreateView event is fired as soon as the control creates a new view. The Items property retrieves the view' items collection. The Columns property retrieves the view's columns collection.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

property CascadeTree.DefColumnWidth as Long

Specifies the width to create a new cascade column.

Type	Description
Long	A long expression that specifies the width to create a new cascade column.

By default, the DefColumnWidth property is 256. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area. The Width property specifies the width of the view. The WidthToFit property specifies the width of the view to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeTree.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 specify the control's foreground color. Use the BackColor property to specify the control's background color. Use the Font property to specify the control's font.

The following VB sample disables the control:

```
CascadeTree1.Enabled = False
```

The following C++ sample disables the control:

```
m_cascadetree.SetEnabled( FALSE );
```

The following VB.NET sample disables the control:

```
AxCascadeTree1.Enabled = False
```

The following C# sample disables the control:

```
axCascadeTree1.Enabled = false;
```

The following VFP sample disables the control:

```
With thisform.CascadeTree1
   .Object.Enabled = False
EndWith
```

method CascadeTree.EndUpdate ()

Resumes painting the control after painting is suspended by the BeginUpdate method.

Type	Description

The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Use BeginUpdate and EndUpdate statement each time when the control requires more changes. Using the BeginUpdate and EndUpdate methods increase the speed of changing the control properties by preventing it from painting during changing.

property CascadeTree.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.

property CascadeTree.ExecuteContextMenu as Long

Executes a command from the object's context menu.

Type	Description
Long	A Long expression that determines the identifier of the command to be executed.

By default, the ExecuteContextMenu property is 0. The ExecuteContextMenu property specifies the identifier of the command to be executed (id option in the ShowContextMenu property). The ExecuteContextMenu property has effect only during the ViewEndChanging event, when the Operation parameter is exExecuteContextMenu(21). The AllowContextMenu property specifies whether the control shows the object's context menu when the user presses the right click over a file or folder.

The following sample shows how you can append new items to the object's context menu and displays a message when a command is selected from the context menu:

```
Private Sub CascadeTree1_StateChange(ByVal State As
EXCASCADETREELibCtl.StateChangeEnum)
   With CascadeTree1
      If (State = ShowContextMenu) Then
         .ShowContextMenu = .ShowContextMenu + ",Item 1[id=1][def],Popup[id=2](Sub-
Item 2[id=2],[sep],Sub-Item 3[id=3])"
      Else
         If (State = ExecuteContextMenu) Then
            Debug.Print "You selected the command: " & .ExecuteContextMenu
         End If
      End If
   End With
End Sub
```

The following sample shows how you can prevent executing a specific command:

```
Private Sub CascadeTree1_StateChange(ByVal State As
EXCASCADETREELibCtl.StateChangeEnum)
   With CascadeTree1
      If (State = ExecuteContextMenu) Then
         If Not (.ExecuteContextMenu = 17) Then   ' Delete
            Debug.Print "You selected the command: " & .ExecuteContextMenu
```

```
        Else
            .ExecuteContextMenu = 0
            MsgBox "Delete is disabled."
        End If
    End If
  End With
End Sub
```

method CascadeTree.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, the following sample retrieves the beginning date (as string) for the default bar in the first visible item:

```
Debug.Print CascadeTree1.ExecuteTemplate("Items.ItemBar(FirstVisibleItem(),``,1)")
```

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.*

property CascadeTree.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. Use the BackColorLevelHeader property to specify the background color of the header when it displays multiple levels.

property CascadeTree.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 FilterBarCaption property to specify the caption of the control's filter bar.

property CascadeTree.FitCascadeColumns as Long

Retrieves or sets a value that indicates the number of cascading columns to fit.

Type	Description
Long	A long expression that indicates the number of cascading columns to fit.

By default, the FitCascadeColumns property is 4. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns. The Width property specifies the width of the view. The WidthToFit property specifies the width of the view to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

method CascadeTree.FitToClient ([FitColumnsCount as Variant])

Resizes or/and moves the all cascade columns to fit the control's client area.

Type	Description
FitColumnsCount as Variant	A long expression that specifies the number of columns to fit. If missing, it indicates 4.

The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The Width property specifies the width of the view. The WidthToFit property specifies the width of the view to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeTree.Font as IFontDisp

Retrieves or sets the Font object used to paint control.

Type	Description
IFontDisp	A Font object being used to paint the items within the control.

Use the Font property to change the control's font. Use the Refresh method to refresh the control.

The following VB sample assigns by code a new font to the control:

```
With CascadeTree1
   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_cascadetree.GetFont();
font.SetName( "Tahoma" );
m_cascadetree.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 AxCascadeTree1
   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);
axCascadeTree1.Font = font;
```

```
axCascadeTree1.CtlRefresh();
```

The following VFP sample assigns by code a new font to the control:

```
with thisform.CascadeTree1.Object
   .Font.Name = "Tahoma"
   .Refresh()
endwith
```

property CascadeTree.ForeColor as Color

Retrieves or sets the control's foreground color.

Type	Description
Color	A color expression that indicates the control's foreground color.

Use the ForeColor property to change the control's foreground color. Use the ForeColorAlternate property specifies the control's alternate foreground color. Use the BackColor / BackColorAlternate property to specify the control's background color. The Background property returns or sets a value that indicates the background color for parts in the control. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items.

property CascadeTree.ForeColorAlternate as Color

Specifies the control's alternate foreground color.

Type	Description
Color	A Color expression that specifies the view's foreground color for an alternate view.

The ForeColorAlternate property specifies the control's alternate foreground color. Use the ForeColor property to change the control's foreground color.

property CascadeTree.ForeColorHeader as Color

Specifies the header's foreground color.

Type	Description
Color	A color expression that indicates the foreground color of the control's header bar.

Use the BackColorHeader and ForeColorHeader properties to define colors used to paint the control's header bar. Use the HeaderVisible property to show or hide the control's header. Use the HeaderHeight property to specify the height of the control's header bar. Use the LevelKey property to allow multiple levels header bar.

property CascadeTree.ForeColorSortBar as Color

Retrieves or sets a value that indicates the sort bar's foreground color.

Type	Description
Color	A color expression that indicates the foreground color of the control's sort bar.

Use the ForeColorSortBar property to specify the foreground color of the caption in the control's sort bar. Use the SortBarVisible property to show the control's sort bar. Use the SortBarCaption property to specify the caption of the sort bar, when the control's sort bar contains no columns. Use the BackColorSortBar property to specify the background color of the control's sort bar. Use the BackColorSortBarCaption property to specify the caption's background color in the control's sort bar. Use the ForeColor property to specify the control's foreground color. Use the ForeColorHeader property to specify the background color of the control's header bar.

method CascadeTree.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 CascadeTree.

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 CascadeTree.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. 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.

method CascadeTree.FreezeEvents (Freeze as Boolean)

Prevents the control to fire any event.

Type	Description
Freeze as Boolean	A Boolean expression that specifies whether the control' events are froze or unfroze

The FreezeEvents(True) method freezes the control's events until the FreezeEvents(False) method is called.

property CascadeTree.HeaderAppearance as AppearanceEnum

Retrieves or sets a value that indicates the header's appearance.

Type	Description
AppearanceEnum	A boolean expression that specifies the appearance of the columns header.

Use the HeaderAppearance property to define the appearance of the columns header bar. The user can't resize the columns at runtime, if the HeaderAppearance property is None2. Use the ColumnsAllowSizing property to allow resizing the columns, when the control's header bar is not visible. Use the Appearance property to define the control's appearance. Use the HeaderVisible property to hide the control's header bar.

property CascadeTree.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 Visible property to hide a particular column. Use the ColumnFromPoint property to access the column from point. If the control's header bar is hidden, the ColumnFromPoint property returns -1. Use the LevelKey property to allow multiple levels header bar. Use the FormatLevel property to display multiple levels in the column's header. 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. Use the AllowSizing property to disable resizing a column when user clicks the right margin of the column. Use the ColumnsAllowSizing property to allow resizing the columns, when the control's header bar is not visible. The Background(exCursorHoverColumn) property specifies the visual appearance of the column's header when the cursor hovers it.

property CascadeTree.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).

The following sample shows how to put a custom size picture in the column's header:

```
<CONTROL>.HTMLPicture("pic1") = "c:/temp/editors.gif"
<CONTROL>.HTMLPicture("pic2") = "c:/temp/editpaste.gif"

<COLUMN1>.HTMLCaption = "A <img>pic1</img>"
<COLUMN2>.HTMLCaption = "B <img>pic2</img>"
<COLUMN3>.HTMLCaption = "A <img>pic1</img> + B <img>pic2</img>"
```

property CascadeTree.hWnd as Long

Retrieves the control's window handle.

Type	Description
Long	A long expression that indicates the handle of the control's window.

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 CascadeTree.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. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection.

property CascadeTree.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

property CascadeTree.ItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as HITEM

Retrieves the item 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 Y location of the mouse pointer. The y values is always expressed in client coordinates.
HITEM	A long expression that specifies the handle of the item from the cursor.

The ItemFromPoint property retrieves the item from the point. The ItemFromPoint(-1,-1) property retrieves and item from the current cursor position. The ViewFromPoint property retrieves the view from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

property CascadeTree.Layout as String

Saves or loads the control's layout, such current selection for each panel, the widths of the cascade columns, and so on.

Type	Description
String	A String expression that defines the current's layout.

The Layout property saves or loads the control's layout, such current selection for each panel, the widths of the cascade columns, and so on. For instance, you can use the Layout property to save and restore later the current views, which includes the selection in each panel, and so on.

property CascadeTree.MaxColumnWidth as Long

Specifies the maximum width for any cascade column.

Type	Description
Long	A Long expression that specifies the maximum width for any cascade column.

By default, the MaxColumnWidth property is -1. While negative, there is no upper limit, so the MaxColumnWidth property has no effect. The Mode property indicates the mode the control displays the cascade columns. The DefColumnWidth property specifies the width to create a new cascade column. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area. The Width property specifies the width of the view. The WidthToFit property specifies the width of the view to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeTree.MinColumnWidth as Long

Specifies the minimum width for any cascade column.

Type	Description
Long	A Long expression that specifies the minimum width for any cascade column.

By default, the MinColumnWidth property is 32. The Mode property indicates the mode the control displays the cascade columns. The DefColumnWidth property specifies the width to create a new cascade column. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area. The Width property specifies the width of the view. The WidthToFit property specifies the width of the view to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeTree.Mode as CascadeModeEnum

Indicates the mode the control displays the cascade columns.

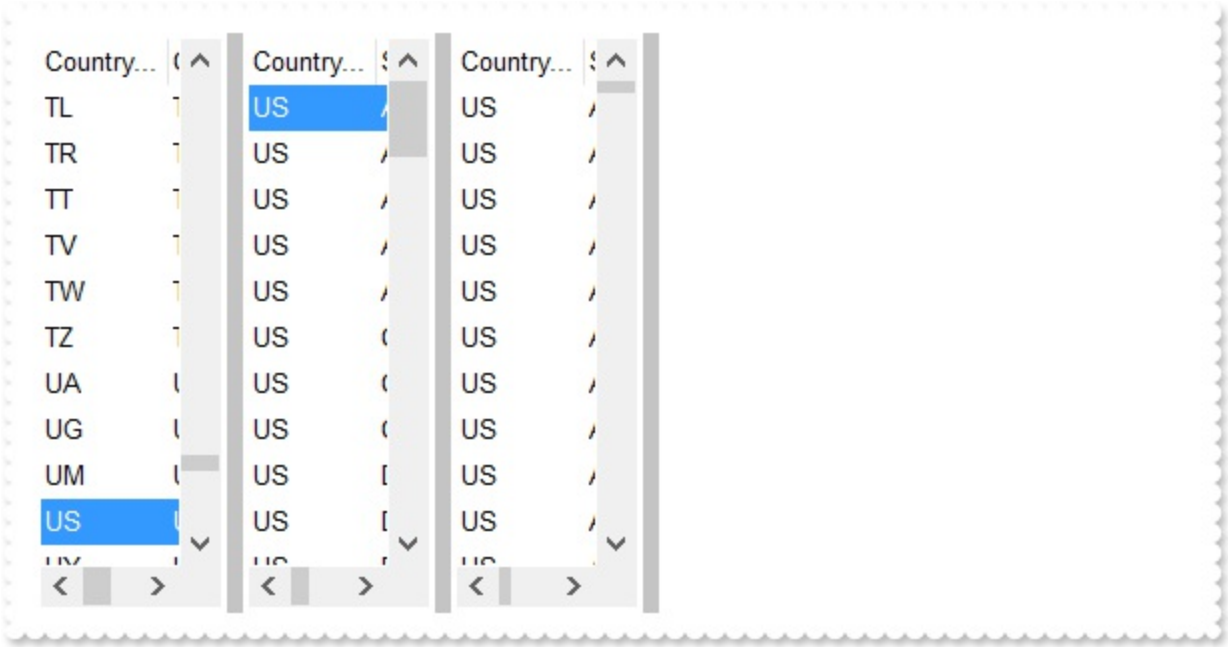
Type	Description
CascadeModeEnum	A CascadeModeEnum expression that indicates how the control displays the cascade columns once a file or more are selected.

By default, the Mode property is exSplitFixCascadeMode | exAutoFitOnResizeClient. The Mode property indicates the mode the control displays the cascade columns. The DefColumnWidth property specifies the width to create a new cascade column. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area.

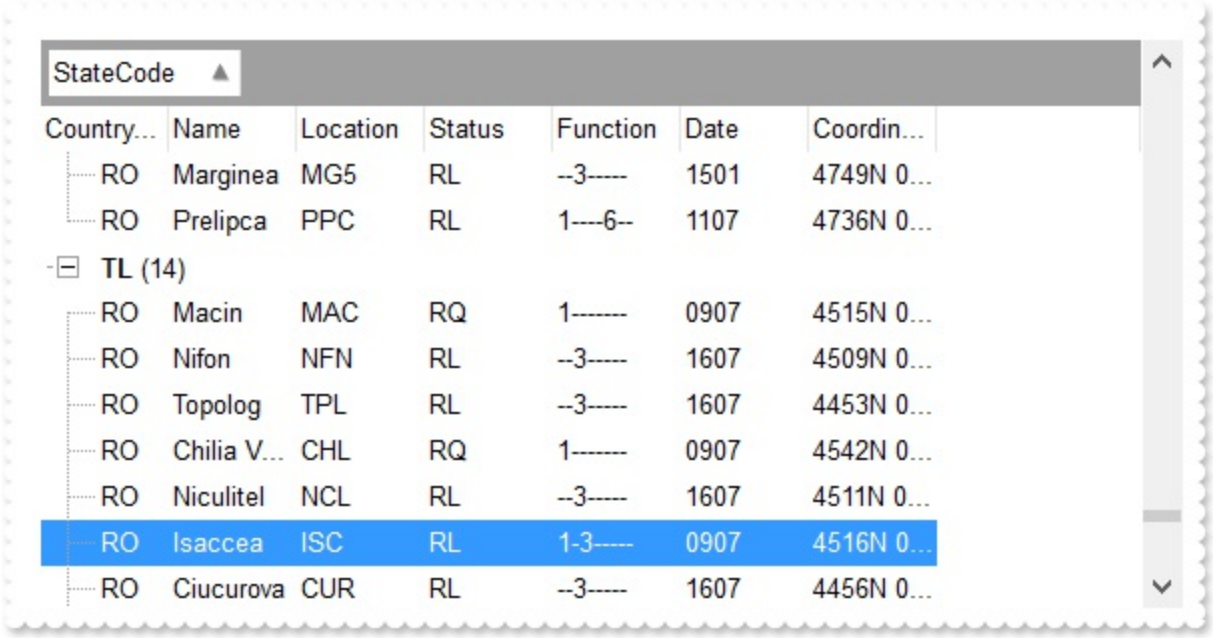
The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

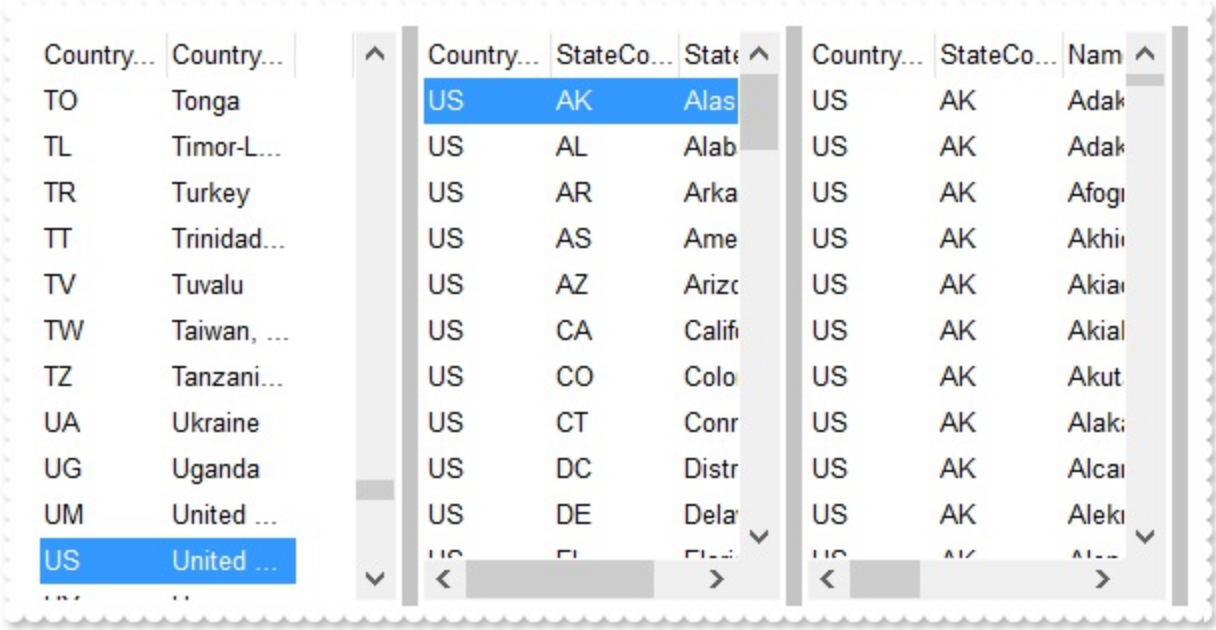
The following screen shot shows the control while the Mode property includes exFixCascadeMode:



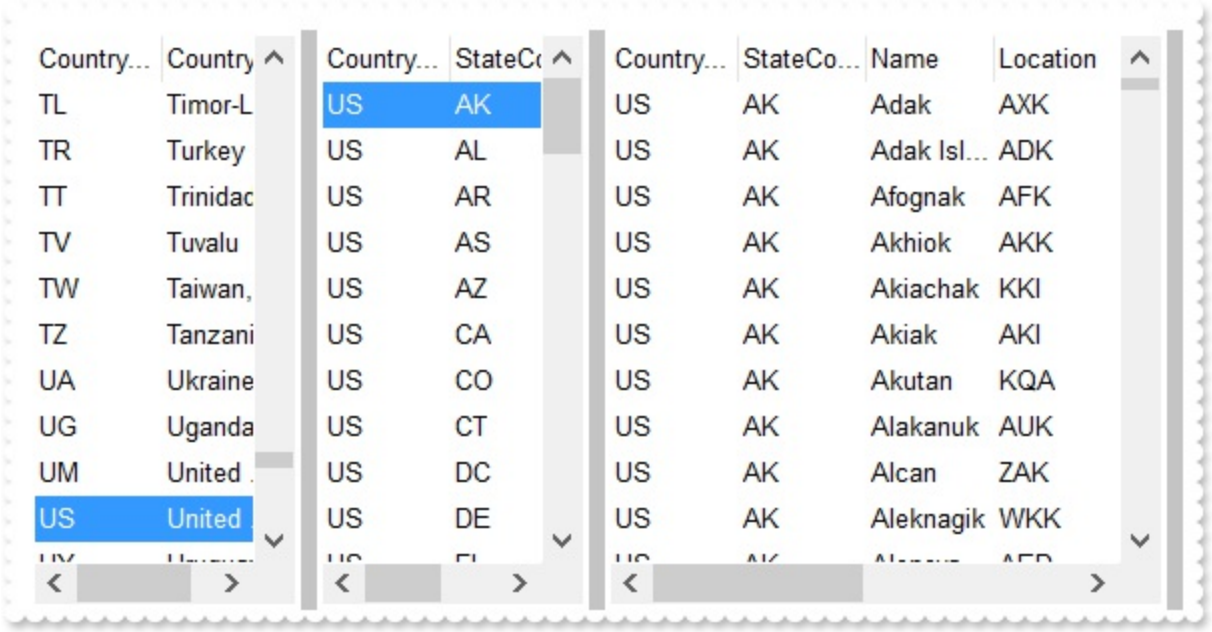
The following screen shot shows the control while the Mode property includes exSingleCascadeMode:



The following screen shot shows the control while the Mode property includes exSplitEqualCascadeMode:



The following screen shot shows the control while the Mode property includes exSplitFixCascadeMode:


Country... Country
TL Timor-L
TR Turkey
TT Trinidac
TV Tuvalu
TW Taiwan,
TZ Tanzani
UA Ukraine
UG Uganda
UM United
US United
Country... StateCo
US AK
US AL
US AR
US AS
US AZ
US CA
US CO
US CT
US DC
US DE
Country... StateCo... Name Location
US AK Adak AXK
US AK Adak Isl... ADK
US AK Afognak AFK
US AK Akhiok AKK
US AK Akiachak KKI
US AK Akiak AKI
US AK Akutan KQA
US AK Alakanuk AUK
US AK Alcan ZAK
US AK Aleknagik WKK

property CascadeTree.Name as String

Selects the path using the name for each view.

Type	Description
String	A String expression that indicates the path of selected items

The Name property is similar with the Select property, excepts it uses the Name column to build the path. The Name property of each View object specifies the index or the caption of the column that determines the name of the view. The Name property can select items using wild characters such as * or ?, if the view's SingleSel property is False.

property CascadeTree.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.

Use the Picture property to load a picture on the control's background. By default, the control has no picture associated. Use the PictureDisplay property to layout the control's picture on the control's background. Use the BackColor property to specify the control's background color. Use the ForeColor property to change the control's foreground color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items.

property CascadeTree.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. Use 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. Use the ForeColor property to change the control's foreground color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items

method CascadeTree.Refresh ()

Refreses the control.

Type	Description

method CascadeTree.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 = CascadeTree1.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 = CascadeTree1.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:

```
 CascadeTree1.ReplaceIcon 0,  i, i specifies the index of icon removed.
```

The following VB clears the control's icons collection:

```
 CascadeTree1.ReplaceIcon 0,  -1
```

property CascadeTree.ScrollButtonHeight as Long

Specifies the height of the button in the vertical scrollbar.

Type	Description
Long	A long expression that defines the height of the button in the vertical scroll bar.

By default, the ScrollButtonHeight property is -1. If the ScrollButtonHeight property is -1, the control uses the default height (from the system) for the buttons in the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeTree.ScrollButtonWidth as Long

Specifies the width of the button in the horizontal scrollbar.

Type	Description
Long	A long expression that defines the width of the button in the horizontal scroll bar.

By default, the ScrollButtonWidth property is -1. If the ScrollButtonWidth property is -1, the control uses the default width (from the system) for the buttons in the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeTree.ScrollFont (ScrollBar as ScrollBarEnum) as IFontDisp

Retrieves or sets the scrollbar's font.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
IFontDisp	A Font object

Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar.

property CascadeTree.ScrollHeight as Long

Specifies the height of the horizontal scrollbar.

Type	Description
Long	A long expression that defines the height of the horizontal scroll bar.

By default, the ScrollHeight property is -1. If the ScrollHeight property is -1, the control uses the default height of the horizontal scroll bar from the system. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeTree.ScrollOrderParts(ScrollBar as ScrollBarEnum) as String

Specifies the order of the buttons in the scroll bar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the order of buttons is displayed.
String	A String expression that indicates the order of the buttons in the scroll bar. The list includes expressions like l, l1, ..., l5, t, r, r1, ..., r6 separated by comma, each expression indicating a part of the scroll bar, and its position indicating the displaying order.

Use the ScrollOrderParts to customize the order of the buttons in the scroll bar. By default, the ScrollOrderParts property is empty. If the ScrollOrderParts property is empty the default order of the buttons in the scroll bar are displayed like follows:

so, the order of the parts is: l1, l2, l3, l4, l5, l, t, r, r1, r2, r3, r4, r5 and r6. Use the ScrollPartVisible to specify whether a button in the scrollbar is visible or hidden. Use the ScrollPartEnable property to enable or disable a button in the scroll bar. Use the ScrollPartCaption property to assign a caption to a button in the scroll bar.

Use the ScrollOrderParts property to change the order of the buttons in the scroll bar. For instance, "l,r,t,l1,r1" puts the left and right buttons to the left of the thumb area, and the l1 and r1 buttons right after the thumb area. If the parts are not specified in the ScrollOrderParts property, automatically they are added to the end.

The list of supported literals in the ScrollOrderParts property is:

- **l** for exLeftBPart, (<) The left or top button.
- **l1** for exLeftB1Part, (L1) The first additional button, in the left or top area.
- **l2** for exLeftB2Part, (L2) The second additional button, in the left or top area.
- **l3** for exLeftB3Part, (L3) The third additional button, in the left or top area.
- **l4** for exLeftB4Part, (L4) The forth additional button, in the left or top area.
- **l5** for exLeftB5Part, (L5) The fifth additional button, in the left or top area.
- **t** for exLowerBackPart, exThumbPart and exUpperBackPart, The union between the exLowerBackPart and the exUpperBackPart parts.
- **r** for exRightBPart, (>) The right or down button.

- **r1** for exRightB1Part, (R1) The first additional button in the right or down side.
- **r2** for exRightB2Part, (R2) The second additional button in the right or down side.
- **r3** for exRightB3Part, (R3) The third additional button in the right or down side.
- **r4** for exRightB4Part, (R4) The forth additional button in the right or down side.
- **r5** for exRightB5Part, (R5) The fifth additional button in the right or down side.
- **r6** for exRightB6Part, (R6) The sixth additional button in the right or down side.

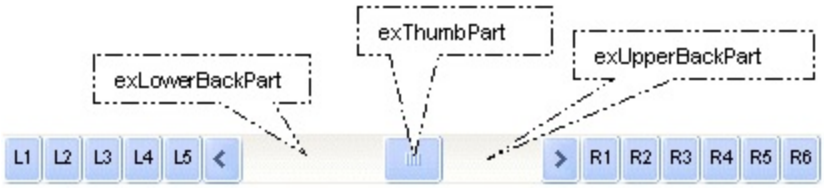
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

property CascadeTree.ScrollPartCaption(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as String

Specifies the caption being displayed on the specified scroll part.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displated
String	A String expression that specifies the caption being displayed on the part of the scroll bar.

Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.



By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

The following VB sample adds up and down additional buttons to the control's vertical scroll bar :

```
With CascadeTree1
  .BeginUpdate
    .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
```

```
    .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
    .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
  .EndUpdate
End With
```

The following VB.NET sample adds up and down additional buttons to the control's vertical scroll bar :

```
With AxCascadeTree1
  .BeginUpdate()
  .set_ScrollPartVisible(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part Or
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, True)
  .set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
  .set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, "<img></img>2")
  .EndUpdate()
End With
```

The following C# sample adds up and down additional buttons to the control's vertical scroll bar :

```
axCascadeTree1.BeginUpdate();
axCascadeTree1.set_ScrollPartVisible(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part |
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, true);
axCascadeTree1.set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axCascadeTree1.set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axCascadeTree1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical scroll bar :

```
m_cascadetree.BeginUpdate();
m_cascadetree.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
```

```
m_cascadetree.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img></img>1") );
m_cascadetree.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img></img>2") );
m_cascadetree.EndUpdate();
```

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

```
With thisform.CascadeTree1
   .BeginUpdate
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

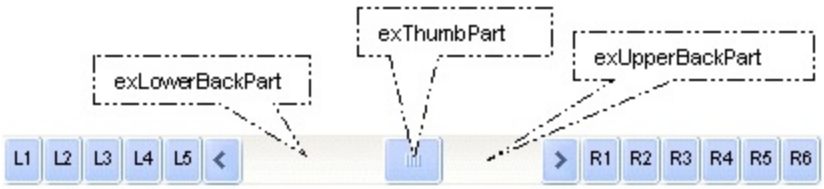
```
*** ActiveX Control Event ***
LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))
```

property CascadeTree.ScrollPartEnable(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is enabled or disabled.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is enabled or disabled.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll bar being enabled or disabled.
Boolean	A Boolean expression that specifies whether the scrollbar's part is enabled or disabled.

By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.

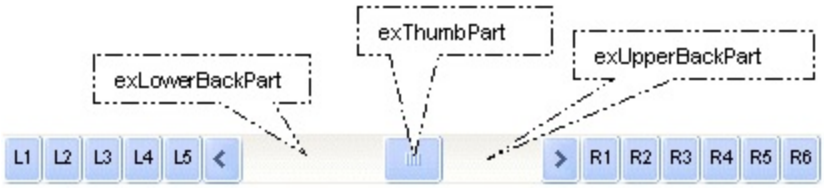


property CascadeTree.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is visible or hidden.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is visible or hidden.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll bar being visible
Boolean	A Boolean expression that specifies whether the scrollbar's part is visible or hidden.

Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the Background property to change the visual appearance for any part in the control's scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.



By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

The following VB sample adds up and down additional buttons to the control's vertical scroll bar :

```
With CascadeTree1
  .BeginUpdate
    .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
```

```
    .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
    .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
  .EndUpdate
End With
```

The following VB.NET sample adds up and down additional buttons to the control's vertical scroll bar :

```
With AxCascadeTree1
  .BeginUpdate()
  .set_ScrollPartVisible(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part Or
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, True)
  .set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
  .set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, "<img></img>2")
  .EndUpdate()
End With
```

The following C# sample adds up and down additional buttons to the control's vertical scroll bar :

```
axCascadeTree1.BeginUpdate();
axCascadeTree1.set_ScrollPartVisible(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part |
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, true);
axCascadeTree1.set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axCascadeTree1.set_ScrollPartCaption(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
EXEXCASCADETREELib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axCascadeTree1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical scroll bar :

```
m_cascadetree.BeginUpdate();
m_cascadetree.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
```

```
m_cascadetree.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img></img>1") );
m_cascadetree.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img></img>2") );
m_cascadetree.EndUpdate();
```

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

```
With thisform.CascadeTree1
   .BeginUpdate
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

```
*** ActiveX Control Event ***
LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))
```

property CascadeTree.ScrollThumbSize(ScrollBar as ScrollBarEnum) as Long

Specifies the size of the thumb in the scrollbar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
Long	A long expression that defines the size of the scrollbar's thumb.

Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb. By default, the ScrollThumbSize property is -1, that makes the control computes automatically the size of the thumb based on the scrollbar's range. If case, use the fixed size for your thumb when you change its visual appearance using the Background(exVSThumb) or Background(exHSThumb) property. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar.

property CascadeTree.ScrollToolTip(ScrollBar as ScrollBarEnum) as String

Specifies the tooltip being shown when the user moves the scroll box.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical scroll bar or the horizontal scroll bar.
String	A string expression being shown when the user clicks and moves the scrollbar's thumb.

Use the ScrollToolTip property to specify whether the control displays a tooltip when the user clicks and moves the scrollbar's thumb. By default, the ScrollToolTip property is empty. If the ScrollToolTip property is empty, the tooltip is not shown when the user clicks and moves the thumb of the scroll bar. Use the SortPartVisible property to specify the parts being visible in the control's scroll bar.

The following VB sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
Private Sub CascadeTree1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As
Long)
   If (Not Horizontal) Then
      CascadeTree1.ScrollToolTip(exVScroll) = "Record " & NewVal
   End If
End Sub
```

The following VB.NET sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
Private Sub AxCascadeTree1_OffsetChanged(ByVal sender As System.Object, ByVal e As
AxEXEXCASCADETREELib._ICascadeTreeEvents_OffsetChangedEvent) Handles
AxCascadeTree1.OffsetChanged
   If (Not e.horizontal) Then
      AxCascadeTree1.set_ScrollToolTip(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
"Record " & e.newVal.ToString())
   End If
End Sub
```

The following C++ sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
void OnOffsetChangedCascadeTree1(BOOL Horizontal, long NewVal)
{
   if ( !Horizontal )
   {
      CString strFormat;
      strFormat.Format( _T("%i"), NewVal );
      m_cascadetree.SetScrollToolTip( 0, strFormat );
   }
}
```

The following C# sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
private void axCascadeTree1_OffsetChanged(object sender,
AxEXEXCASCADETREELib._ICascadeTreeEvents_OffsetChangedEvent e)
{
   if ( !e.horizontal )
      axCascadeTree1.set_ScrollToolTip(EXEXCASCADETREELib.ScrollBarEnum.exVScroll,
"Record " + e.newVal.ToString());
}
```

The following VFP sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
*** ActiveX Control Event ***
LPARAMETERS horizontal, newval

If (1 # horizontal) Then
   thisform.CascadeTree1.ScrollToolTip(0) = "Record " + ltrim(str(newval))
EndIf
```

property CascadeTree.ScrollWidth as Long

Specifies the width of the vertical scrollbar.

Type	Description
Long	A long expression that defines the width of the vertical scroll bar.

By default, the ScrollWidth property is -1. If the ScrollWidth property is -1, the control uses the default width of the vertical scroll bar from the system. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeTree.SelBackColor as Color

Retrieves or sets a value that indicates the selection background color.

Type	Description
Color	A color expression that indicates the background color for selected items. 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 SelBackColor property specifies the background color for selected items. Use the SelForeColor property to specify the foreground color for selected items. Use the BackColor property to specify the control's background color. Use the ForeColor property to specify the control's foreground color.

property CascadeTree.Select as String

Selects the path using the key for each view.

Type	Description
String	A String expression that defines the path of selected items, using the Key column in each view.

The Key property indicates the column that defines the key of the view. Based on the key, and the current selection the next view is created. The Select property can select items using wild characters such as * or ?, if the view's SingleSel property is False. The Key property can be specified also through Key field of the control's DataSource property. The view's Select property selects items within the view and its descendents.

property CascadeTree.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.

The SelForeColor property specifies the foreground color for selected items. Use the SelBackColor property to specify the background color for selected items. Use the ForeColor property to specify the control's foreground color. Use the BackColor property to specify the control's background color.

property CascadeTree.ShowContextMenu as String

Specifies the object's context menu.

Type	Description
String	A String expression that specifies the commands to be displayed in the object's context menu. The ShowContextMenu property supports expressions, so you can combine the default context menu, with your own context menu for any file/folder.

By default, the ShowContextMenu property is empty. The ShowContextMenu property can be used to disable, update, remove or add new items. The ShowContextMenu property indicates the items to be displayed on the object's context menu. The AllowContextMenu property specifies whether the control shows the object's context menu when the user presses the right click over a file or folder.

The ShowContextMenu property supports the following predefined keywords:

- **view** keyword indicates the view the context menu is displayed for
- **hlevel** keyword specifies the index of the horizontal cascade column view the context menu is displayed for.
- **hlevels** keyword gets the count of horizontal cascade columns
- **vlevel** keyword specifies index of the vertical splitting panel the context menu is displayed for
- **vlevels** keyword gets the count of vertically split panels

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

The ShowContextMenu property indicates the list of commands to be displayed in the context menu, separated by comma (,). Each command must have an id parameter, that specifies the identifier of the command. Optional parameters are def for default item, and dis for disabled items. The sep parameter indicates a separator item. If adding new items to the object's context menu, use the ExecuteContextMenu property to get the identifier of the command to be executed during the ViewEndChanging event, when the Operation parameter is exExecuteContextMenu(21).

For instance, the ShowContextMenu property on *"Item 1[id=1][def],Popup[id=2](Sub-Item 2[id=2],[sep],Sub-Item 3[id=3])"* shows the context menu as following:

property CascadeTree.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 RepalceIcon method to add, remove or clear icons in the control's images collection.

method CascadeTree.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 $_{subscript}$ 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:

- **<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 CascadeTree.SplitViewHeight as String

Specifies the height of split panels, separated by comma.

Type	Description
String	A String expression that specifies the height for each vertical split-panel, separated by comma character.

By default, the SplitViewHeight property is "", so no additional view is displayed. The SplitViewHeight property specifies the height of split panels, separated by comma. The AllowSplitView property specifies whether the user can split the control into multiple-views. The Background(exHSplitBar) property specifies the visual appearance of the control's split bar (horizontal split bar)

property CascadeTree.StatusBarHeight as Long

Specifies the height of the control's status bar.

Type	Description
Long	A long expression that specifies the height of the control's status bar.

By default, the StatusBarHeight property is -1, which specifies that status bar height is automatically computed based on its label. The StatusBarHeight property specifies the height of the control's status bar. The StatusBarLabel property specifies the HTML label the control's status bar is displaying. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The HeaderVisible property retrieves or sets a value that indicates whether the control's header bar is visible or hidden. The Background(exStatusBackColor) / Background(exStatusForeColor) specifies the status bar's background / foreground color. The Background(exStatusPanelBackColor) specifies the status panel's background color.

property CascadeTree.StatusBarLabel as String

Specifies the HTML label the control's status bar is displaying.

Type	Description
String	A String expression that specifies the HTML label the control's status bar is displaying.

By default, the StatusBarLabel property is "". The StatusBarLabel property specifies the HTML label the control's status bar is displaying. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The HeaderVisible property retrieves or sets a value that indicates whether the control's header bar is visible or hidden. The Background(exStatusBackColor) / Background(exStatusForeColor) specifies the status bar's background / foreground color. The Background(exStatusPanelBackColor) specifies the status panel's background color. The StatueBarHeight property Specifies the height of the control's status bar.

The StatusBarLabel property 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 $^{\text{subscript}}$
- **<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:

- **<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 CascadeTree.StatusBarVisible as StatusBarAnchorEnum

Specifies whether the control's status bar is visible or hidden.

Type	Description
StatusBarAnchorEnum	A StatusBarAnchorEnum expression that specifies whether the control's status bar is visible or hidden.

By default, the StatusBarVisible property is exStatusBarAnchorTop, and so the status bar is visible and aligned to the top side of the control. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying. The HeaderVisible property retrieves or sets a value that indicates whether the control's header bar is visible or hidden. The Background(exStatusBackColor) / Background(exStatusForeColor) specifies the status bar's background / foreground color. The Background(exStatusPanelBackColor) specifies the status panel's background color.

property CascadeTree.Template as String

Specifies the control's template.

Type	Description
String	A string expression that indicates 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 execute a template script and gets the result.

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*

property CascadeTree.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 CascadeTree.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 CascadeTree.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. 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

property CascadeTree.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.

property CascadeTree.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.

property CascadeTree.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.

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.

property CascadeTree.UseTabKey as Boolean

Retrieves or sets a value that specifies whether the Tab or SHIFT + Tab key navigates through the cascading columns.

Type	Description
Boolean	A Boolean expression that specifies whether Tab or SHIFT + Tab key navigates through the cascading columns.

By default, the UseTabKey property is True, which indicates that TAB activates the next cascade column, based on the current selection. The UseTabKey property specifies whether the Tab or SHIFT + Tab key navigates through the cascading columns.

property CascadeTree.Version as String

Retrieves the control's version.

Type	Description
String	A string expression that indicates the control's version.

The Version property is read-only. The Version property specifies the version of the control you are running.

property CascadeTree.View as View

Returns the view you are currently working on.

Type	Description
View	A View object that specifies the view where the event occurs.

The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs. The CreateView event is fired as soon as the control creates a new view. The DefaultView property specifies the default view on the control. The Items property retrieves the view' items collection. The Columns property retrieves the view's columns collection.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

property CascadeTree.ViewColumnFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, ppView as View) as Long

Retrieves the view and 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 Y location of the mouse pointer. The y values is always expressed in client coordinates.
ppView as View	A View object from the cursor.
Long	A Long expression that specifies the index of the column from the cursor.

The ViewColumnFromPoint property retrieves the view and column from the point. The ViewColumnFromPoint(-1,-1) property retrieves the view and column from the current cursor position. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ColumnFromPoint property retrieves the column from the point.

property CascadeTree.ViewFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as View

Retrieves the view 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 Y location of the mouse pointer. The y values is always expressed in client coordinates.
View	A View object from the cursor.

The ViewFromPoint property retrieves the view from the point. The ViewFromPoint(-1,-1) property retrieves the view from the current cursor position. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

property CascadeTree.ViewItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, ppView as View, ColIndex as Long, HitTestInfo as HitTestInfoEnum) as HITEM

Retrieves the view and item 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 Y location of the mouse pointer. The y values is always expressed in client coordinates.
ppView as View	A View object from the cursor.
ColIndex as Long	A Long expression that specifies the index of the column from the cursor.
HitTestInfo as HitTestInfoEnum	A HitTestInfoEnum expression that indicates the hit test code.
HITEM	A long expression that specifies the handle of the item from the cursor.

The ViewItemFromPoint property retrieves the view and item from the point. The ViewItemFromPoint(-1,-1) property retrieves the view and item from the current cursor position. The ViewFromPoint property retrieves the view from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

property CascadeTree.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.

Column object

The ExCascadeTree control supports multiple columns. The Columns object contains a collection of Column objects. By default, the control doesn't add any default column, so the user has to add at least one column, before inserting any new items. The Column object supports the following properties and methods:

Name	Description
Alignment	Specifies the column's alignment.
AllowDragging	Retrieves or sets a value indicating whether the user will be able to drag the column.
AllowGroupBy	Specifies if the column can be grouped by.
AllowSizing	Retrieves or sets a value indicating whether the user will be able to change the width of the visible columns by dragging.
AllowSort	Returns or sets a value that indicates whether the user can sort the column by clicking the column's header.
AutoSearch	Specifies the kind of searching while user types characters within the columns.
AutoWidth	Computes the column's width required to fit the entire column's content.
Caption	Retrieves or sets the text displayed to the column's header.
ComputedField	Retrieves or sets a value that indicates the formula of the computed column.
CustomFilter	Retrieves or sets a value that indicates the list of custom filters.
Data	Associates an extra data to the column.
Def	Retrieves or sets a value that indicates the default value of given properties for all cells in the same column.
DefaultSortOrder	Specifies whether the default sort order is ascending or descending.
DisplayExpandButton	Shows or hides the expanding/collapsing button in the column's header.
DisplayFilterButton	Specifies whether the column's header displays the filter button.
	Specifies whether the drop down filter window displays a

DisplayFilterDate	date selector to specify the interval dates to filter for.
DisplayFilterPattern	Specifies whether the dropdown filterbar contains a textbox for editing the filter as pattern.
DisplaySortIcon	Retrieves or sets a value indicating whether the sort icon is visible on column's header, while the column is sorted.
Enabled	Returns or sets a value that determines whether a column's header can respond to user-generated events.
ExpandColumns	Specifies the list of columns to be shown when the current column is expanded.
Expanded	Expands or collapses the column.
Filter	Specifies the column's filter when the filter type is exFilter, exPattern, exDate, exNumeric, exCheck or exImage
FilterBarDropDownWidth	Specifies the width of the drop down filter window proportionally with the width of the column.
FilterList	Specifies whether the drop down filter list includes visible or all items.
FilterOnType	Filters the column as user types characters in the drop down filter window.
FilterType	Specifies the column's filter type.
FormatColumn	Specifies the format to display the cells in the column.
FormatLevel	Retrieves or sets a value that indicates the layout of columns being displayed in the column's header.
GroupByFormatCell	Indicates the format of the cell to be displayed when the column gets grouped by.
GroupByTotalField	Indicates the aggregate formula to be displayed when the column gets grouped by.
HeaderAlignment	Specifies the alignment of the column's caption.
HeaderBold	Retrieves or sets a value that indicates whether the column's caption should appear in bold.
HeaderImage	Retrieves or sets a value indicating the index of an Image in the Images collection, which is displayed to the column's header.
HeaderImageAlignment	Retrieves or sets the alignment of the image in the column's header.
HeaderItalic	Retrieves or sets a value that indicates whether the

	column's caption should appear in italic.
HeaderStrikeOut	Retrieves or sets a value that indicates whether the column's caption should appear in strikeout.
HeaderUnderline	Retrieves or sets a value that indicates whether the column's caption should appear in underline.
HeaderVertical	Specifies whether the column's header is vertically displayed.
HTMLCaption	Retrieves or sets the text in HTML format displayed in the column's header.
Index	Returns a value that represents the index of an object in a collection.
Key	Retrieves or sets a the column's key.
LevelKey	Retrieves or sets a value that indicates the key of the column's level.
MaxWidthAutoResize	Retrieves or sets a value that indicates the maximum column's width when the WidthAutoResize is True.
MinWidthAutoResize	Retrieves or sets a value that indicates the minimum column's width when the WidthAutoResize is True.
PartialCheck	Specifies whether the column supports partial check feature.
Position	Retrieves or sets a value that indicates the position of the column in the header bar area.
Selected	Retrieves or sets a value that indicates whether the cell in the column is selected.
ShowFilter	Shows the column's filter window.
SortOrder	Specifies the column's sort order.
SortPosition	Returns or sets a value that indicates the position of the column in the sorting columns collection.
SortType	Returns or sets a value that indicates the way a control sorts the values for a column.
ToolTip	Specifies the column's tooltip description.
Visible	Retrieves or sets a value indicating whether the column is visible or hidden.
Width	Retrieves or sets the column's width.
WidthAutoResize	Retrieves or sets a value that indicates whether the column is automatically resized according to the width of

the contents within the column.

property Column.Alignment as AlignmentEnum

Retrieves or sets the alignment of the caption in the column's header.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the column's alignment.

Use the Alignment property to align cells in a column.By default the column is left aligned. Use the Alignment property to change the column's alignment. Use the HeaderAlignment property to align the column's caption inside the column's header. By default, all columns are aligned to left. If the column displays the hierarchy lines, and if the Alignment property is RightAlignment the hierarchy lines are painted from right to left side. Use the HasLines property to display the control's hierarchy lines. Use the CellHAlignment property to align a particular cell. Use the HeaderImageAlignment property to align the image in the column's header, if it exists. Use the HeaderImage property to attach an icon to the column's header. Use the Def(exCellDrawPartsOrder) property to specify the order of the drawing parts inside the cell. The RightToLeft property automatically flips the order of the columns.

property Column.AllowDragging as Boolean

Retrieves or sets a value indicating whether the user will be able to drag the column.

Type	Description
Boolean	A boolean expression indicating whether the user will be able to drag the column.

Use the AllowDragging property to forbid user to change the column's position by dragging. If the AllowDragging is false, the column's position cannot be changed by dragging it to another position. Use the AllowSizing property to allow user resizes a column at runtime.

property Column.AllowGroupBy as Boolean

Specifies if the column can be grouped by.

Type	Description
Boolean	A Boolean expression that specifies whether the user can drag and drop the column to be grouped by,

By default, the AllowGroupBy property is True. The AllowGroupBy property has effect only if the control's AllowGroupBy property is True. Use the AllowGroupBy property on False, to prevent a specific column to be sorted/grouped by. The same you can achieve if the AllowSort property is False. The SortBarVisible property specifies whether the control's sort bar is visible or hidden. If the control's sort bar is visible, the user can drag and drop columns to it, so the column get sorted and items grouped. The Group/Ungroup method groups or ungroup the control's list. For instance, you can remove the grouping items, by calling the Ungroup method. The GroupByTotalField property determines the formula to be applied to the column when it gets grouped. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted.

property Column.AllowSizing as Boolean

Retrieves or sets a value indicating whether the user will be able to change the width of the visible column by dragging.

Type	Description
Boolean	A boolean expression that indicates whether the user will be able to change the width of the visible columns by dragging.

Use the AllowSizing property to fix the column's width. Use the ColumnAutoResize property of the control to fit the columns to the control's client area. Use the AllowDragging property to forbid user to change the column's position by dragging. Use the Width property to specify the column's width. Use the ColumnsAllowSizing property to allow resizing the columns, when the control's header bar is not visible. Use the HeaderVisible property to show or hide the control's header bar.

property Column.AllowSort as Boolean

Returns or sets a value that indicates whether the user can sort the column by clicking the column's header.

Type	Description
Boolean	A boolean expression that indicates whether the column gets sorted when the user clicks the column's header.

Sorting by a single column in the control is a simple matter of clicking on the column head. Sorting by multiple columns, however, is not so obvious. But it's actually quite easy. First, sort by the first criterion, by clicking on the column head. Then hold the Shift key down as you click on a second heading. Another option is dragging the column's header to the control's sort bar. The SortBarVisible property shows the control's sort bar. Use the AllowSort property to avoid sorting a column when the user clicks the column's header. Use the SortOnClick property to specify the action that control executes when the user clicks the column's head. Use the AllowDragging property to specify whether the column's header can be dragged. Use the DefaultSortOrder property to specify the column's default sort order, when the user first clicks the column's header. The EnsureOnSort property prevents scrolling the control's content when the user sorts items.

property Column.AutoSearch as AutoSearchEnum

Specifies the kind of searching while user types characters within the columns.

Type	Description
AutoSearchEnum	An AutoSearchEnum expression that defines the type of incremental searching.

By default, the AutoSearch property is exStartWith. The AutoSearch property has effect only if the AutoSearch property of the control is True. Use the AutoSearch property to define a 'contains' incremental search. If the AutoSearch property is exContains, the control searches for items that contains the typed characters. The searching column is defined by the SearchColumnIndex property. Use the ExpandOnSearch property to expand items while user types characters in the control.

property Column.AutoWidth as Long

Computes the column's width required to fit the entire column's content.

Type	Description
Long	A long value that indicates the required width of the column to fit the entire column's content.

Use the AutoWidth property to arrange the columns to fit the entire control's content. The AutoWidth property doesn't change the column's width. Use Width property to change the column's width at runtime. Use the ColumnAutoResize property to specify whether the control resizes all visible columns to fit the control's client area.

property Column.Caption as String

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

Type	Description
String	A string expression that indicates the column's caption.

Each property of Items object that has an argument ColIndex can use the column's caption to identify a column. Adding two columns with the same caption is accepted and these are differentiated by their indexes. Use the HTLMCaption property to display the column's caption using HTML tags. To hide a column use the Visible property of the Column object. Use the HeaderVertical property to display vertically the column's caption. Use the HeaderImage property to assign an icon to a column. The column's caption is displayed using the following font attributes: HeaderBold, HeaderItalic, HeaderUnderline, HeaderStrikeout. Use the Add method to add new columns and to specify their captions. Use the FormatLevel property to display multiple levels in the column's header.

property Column.ComputedField as String

Retrieves or sets a value that indicates the formula of the computed column.

Type	Description
String	A String expression that indicates the formula to compute the field/cell. The formula is applied to all cells in the column with the CellValueFormat property on exText (the exText value is by default).

A computed field or cell displays the result of an arithmetic formula that may include operators, variables and constants. By default, the ComputedField property is empty. If the the ComputedField property is empty, the property have no effect. If the ComputedField property is not empty, all cells in the column, that have the CellValueFormat property on exText, uses the same formula to display their content. For instance, you can use the CellValueFormat property on exHTML, for cells in the column, that need to display other things than column's formula, or you can use the CellValueFormat property on exComputedField, to change the formula for a particular cell. Use the FormatColumn property to format the column.

Use the CellValueFormat property to change the type for a particular cell. Use the CellValue property to specify the cell's content. For instance, if the CellValueFormat property is exComputedField, the Caption property indicates the formula to compute the cell's content.

The Def(exCellValueFormat) property is changed to exComputedField, each time the ComputeField property is changed to a not empty value. If the ComputedField property is set to an empty string, the Def(exCellValueFormat) property is set to exText. Call the Refresh method to force refreshing the control.

The expression may be a combination of variables, constants, strings, dates and operators. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by # character, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM.

Samples:

1. "**1**", the cell displays 1
2. "**%0 + %1**", the cell displays the sum between cells in the first and second columns.
3. "**%0 + %1 - %2**", the cell displays the sum between cells in the first and second columns minus the third column.
4. "**(%0 + %1)*0.19**", the cell displays the sum between cells in the first and second columns multiplied with 0.19.
5. "**(%0 + %1 + %2)/3**", the cell displays the arithmetic average for the first three columns.

6. "**%0 + %1 < %2 + %3**", displays 1 if the sum between cells in the first two columns is less than the sum of third and forth columns.
7. "**proper(%0)**'" formats the cells by capitalizing first letter in each word
8. "**currency(%1)**'" displays the second column as currency using the format in the control panel for money
9. "**len(%0) ? currency(dbl(%0)) : ''**" displays the currency only for not empty/blank cells.
10. "**int(date(%1)-date(%2)) + 'D ' + round(24*(date(%1)-date(%2) - floor(date(%1)-date(%2)))) + 'H''**" displays interval between two dates in days and hours, as xD yH
11. **"2:=((1:=int(0:= date(%1)-date(%0))) = 0 ? '' : str(=:1) + ' day(s)') + (3:=round(24*(=:0-floor(=:0))) ? (len(=:2) ? ' and ' : '') + =:3 + ' hour(s)' : '')"** displays the interval between two dates, as x day(s) [and y hour(s)], where the x indicates the number of days, and y the number of hours. The hour part is missing, if 0 hours is displayed, or nothing is displayed if dates are identical.

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, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

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

property Column.CustomFilter as String

Retrieves or sets a value that indicates the list of custom filters.

Type	Description
String	A String expression that defines the list of custom filters.

By default, the CustomFilter property is empty. The CustomFilter property has effect only if it is not empty, and the FilterType property is not exImage, exCheck or exNumeric. Use the DisplayFilterPattern property to hide the text box to edit the pattern, in the drop down filter window. The All predefined item and the list of custom filter is displayed in the drop down filter window, if the CustomFilter property is not empty. The Blanks and NonBlanks predefined items are not defined, when custom filter is displayed. Use the DisplayFilterButton property to show the button on the column's header to drop down the filter window. Use the Background property to define the visual appearance for the drop down button.

The CustomFilter property defines the list of custom filters as pairs of (caption,pattern) where the caption is displayed in the drop down filter window, and the pattern is get selected when the user clicks the item in the drop down filter window (the FilterType property is set on exPattern, and the Filter property defines the custom pattern being selected). The caption and the pattern are separated by a "||" string (two vertical bars, character 124). The pattern expression may contains multiple patterns separated by a single "|" character (vertical bar, character 124). A pattern may contain the wild card characters '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character. If any of the *, ?, # or | characters are preceded by a \ (escape character) it masks the character itself. If the pattern is not present in the (caption,pattern) pair, the caption is considered as being the pattern too. The pairs in the list of custom patterns are separated by "|||" string (three vertical bars, character 124). So, the syntax of the CustomFilter property should be of: CAPTION [**||** PATTERN [**|** PATTERN]] [**|||** CAPTION [**||** PATTERN [**|** PATTERN]]].

For example, you may have a list of documents and instead of listing the name of each document in the filter drop down list for the names column you may want to list the following:

Excel Spreadsheets
Word Documents
Powerpoint Presentations
Text Documents

And define the filter patterns for each line above as follows:

*.xls
*.doc
*.pps
*.txt, *.log

and so the CustomFilter property should be "**Excel Spreadsheets (*.xls)||*.xls|||Word Documents||*.doc|||Powerpoint Presentations||*.pps|||Text Documents (*.log,*.txt)||*.txt|*.log**". The following screen shot shows this custom filter format.

property Column.Data as Variant

Associates an extra data to the column.

Type	Description
Variant	A Variant expression that indicates the column's extra data.

Use the Data property to assign any extra data to a column. Use the CellData property to assign an extra data to a cell. Use the ItemData property to assign an extra data to an item.

property Column.Def(Property as DefColumnEnum) as Variant

Retrieves or sets a value that indicates the default value of given properties for all cells in the same column.

Type	Description
Property as DefColumnEnum	A DefColumnEnum expression that indicates the property being changed.
Variant	A Variant value that specifies the newly value.

Use the Def property to specify a common value for given properties for all cells in the column. For instance, you can use the Def property to assign check boxes to all cells in the column, without enumerating them.

property Column.DefaultSortOrder as Boolean

Specifies whether the default sort order is ascending or descending.

Type	Description
Boolean	A boolean expression that specifies the default sort order.

Use the DefaultSortOrder property to specify the default sort order, when the column's header is clicked. Use the SortOnClick property to specify when user can sort the columns by clicking the control's header. Use the SortOrder property to sort a column. Use the SortChildren method to sort items at runtime. Use the SingleSort property to allow sorting by multiple columns.

property Column.DisplayExpandButton as Boolean

Shows or hides the expanding/collapsing button in the column's header.

Type	Description
Boolean	A Boolean expression that specifies whether the column's header displays a +/- (expanding button), to let user expands or collapse the column, when it is clicked.

By default, the DisplayExpandButton property is True. The DisplayExpandButton property displays the header's expanding button, only, if it contains child columns specified using the ExpandColumns property. The HasButtons property indicates the way the +/- (expanding button) is shown. Use the DisplayExpandButton property on True and ExpandColumns property to display the columns on multiple levels. The Expanded property expands programmatically a column. The control fires the ViewEndChanging(exLayoutChange) event when the user expands or collapse a column. Use the ExpandItem property to expand or collapse an item. The Index property indicates the column's index. The Visible property specifies whether a column is Visible or hidden.

property Column.DisplayFilterButton as Boolean

Shows or hides the column's filter bar button.

Type	Description
Boolean	A boolean expression that indicates whether the column's filter bar button is visible or hidden.

By default, the DisplayFilterButton property is False. The column's filter button is displayed on the column's caption. *Use the FilterOnType property to enable the Filter-On-Type feature, that allows you to filter the control's data based on the characters you type.*

The DisplayFilterPattern property determines whether the column's filter window includes the "Filter For" (pattern) field. Use the DisplayFilterDate property to include a date selector to the column's drop down filter window. Use the FilterBarDropDownHeight property to specify the height of the drop down filter window. Use the FilterBarDropDownWidth property to specify the width of the drop down filter window. Use the FilterType property to specify the type of the column's filter. Use the FilterType property to filter items based on the caption, check state or icons. Use the FilterList property to specify the list of items being included in the column's drop down filter list. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the Background(exHeaderFilterBarButton) property to change the visual appearance for the drop down filter button. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the CustomFilter property to define you custom filters. Use the ShowFilter method to show programmatically the column's drop down filter window.

property Column.DisplayFilterDate as Boolean

Specifies whether the drop down filter window displays a date selector to specify the interval dates to filter for.

Type	Description
Boolean	A boolean expression that indicates whether the drop down filter window displays a date selector to filter items into a given interval.

By default, the DisplayFilterDate property is False. Use the DisplayFilterDate property to filter items that match a given interval of dates. The DisplayFilterDate property includes a date button to the right of the Date field in the drop down filter window. The DisplayFilterDate property has effect only if the DisplayFilterPattern property is True. If the user clicks the filter's date selector the control displays a built-in calendar editor to help user to include a date to the date field of the drop down filter window. If the Date field in the filter drop down window is not empty, the FilterType property of the Column object is set on exDate, and the Filter property of the Column object points to the interval of dates being used when filtering.

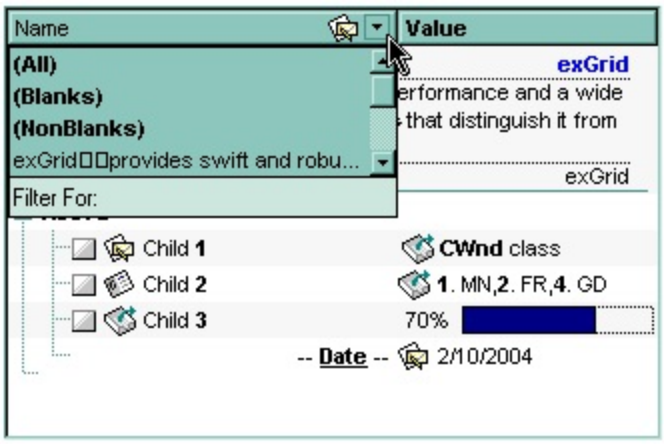
property Column.DisplayFilterPattern as Boolean

Specifies whether the dropdown filter bar contains a textbox for editing the filter as pattern.

Type	Description
Boolean	A boolean expression that indicates whether the pattern field is visible or hidden.

Use the DisplayFilterButton property to show the column's filter button. If the DisplayFilterButton property is False the drop down filter window doesn't include the "Filter For" or "Date" field. Use the DisplayFilterDate property to filter items that match a given interval of dates. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the CustomFilter property to define you custom filters. The "Filter For" (pattern) field in the drop down filter window is always shown if the FilterOnType property is True, no matter of the DisplayFilterPattern property.

The drop down filter window displays the "Filter For" field if the DisplayFilterPattern property is True, and the DisplayFilterDate property is False. If the drop down filter window displays "Filter For" field, and user types the filter inside, the FilterType property of the Column is set to exPattern, and Filter property of the Column object specifies the filter being typed.



The drop down filter window displays the "Date" field if the DisplayFilterPattern property is True, and the DisplayFilterDate property is True. If the drop down filter window displays "Date" field, and user types selects an interval of dates, the FilterType property of the Column is set to exDate, and Filter property of the Column object specifies the interval of dates being used in filtering.

property Column.DisplaySortIcon as Boolean

Retrieves or sets a value indicating whether the sort icon is visible in column's header, while the column is sorted.

Type	Description
Boolean	A boolean expression indicating whether the sort icon is visible on column's header, if the column was sorted by clicking in its header.

Use the DisplaySortIcon property to hide the icon of the column. Use the SortOnClick property to disable sorting columns by clicking in column's header. Use the SortChildren property of the Items object to sort by a column. Use the SortOrder property to sort a column. Use the SingleSort property to allow multiple sort columns.

property Column.Enabled as Boolean

Returns or sets a value that determines whether a column's header can respond to user-generated events.

Type	Description
Boolean	A boolean expression that determines whether a column's header can respond to user-generated events.

Use the Enabled property to disable a column. If a column is disabled, the user can select new items, but any checkbox, radio button, or editor in the cells of the column is disabled. Use the CellEnabled property to disable a particular cell. Use the EnableItem property to disable an item. Use the SelectableItem property to specify the user can select an item.

property Column.ExpandColumns as String

Specifies the list of columns to be shown when the current column is expanded.

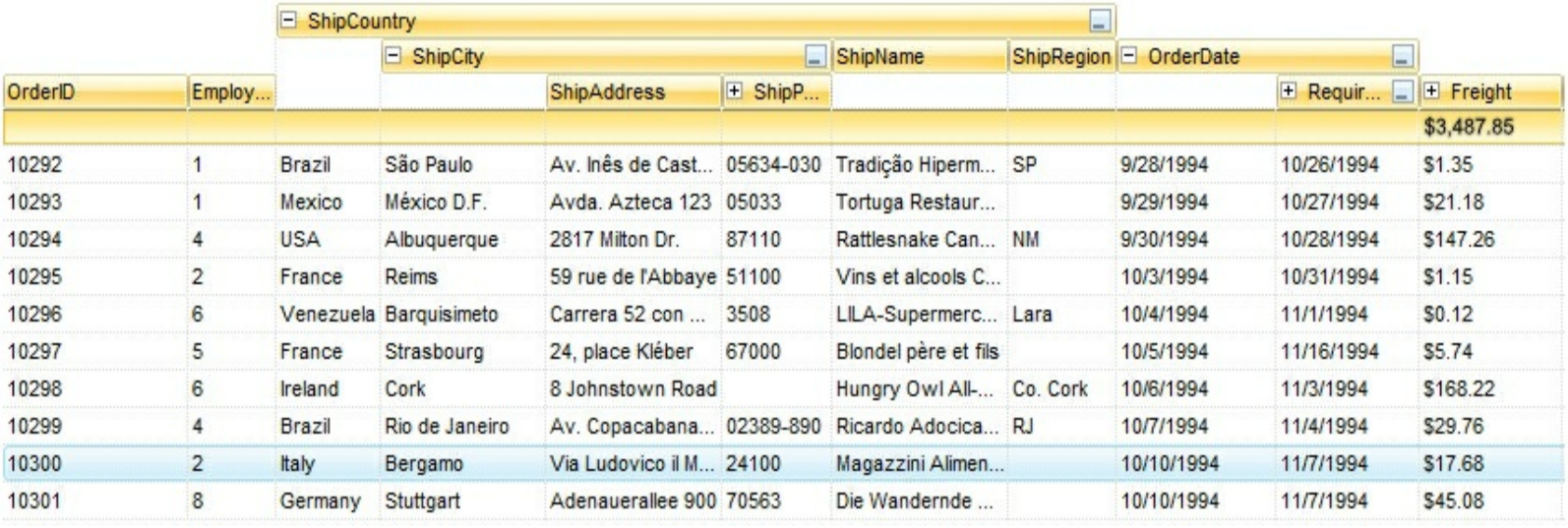
Type	Description
String	A String expression that specifies the list of columns to be shown/hidden when the current column is expanded or collapsed. The list indicates the index of each column to be shown/hidden separated by comma character. For instance, "2,3" indicates that the columns with the index 2 and 3 are displayed bellow the current column.

By default, the ExpandColumns property is empty. Use the ExpandColumns property to display the columns on multiple levels, or to allow your users to expand/collapse the columns. The DisplayExpandButton property specifies whether the column's header displays a +/- (expanding button), to let user expands or collapse the column, when it is clicked. The Expanded property expands programmatically a column. The control fires the ViewEndChanging(exLayoutChange) event when the user expands or collapse a column. Use the ExpandItem property to expand or collapse an item. The Index property indicates the column's index. The Visible property specifies whether a column is Visible or hidden.

The control performs showing/hiding the child columns as follow:

- If the column is expanded, the child columns are shown, and the current column is hidden, if the index of itself it is not included in the ExpandColumns property.
- If the column is collapsed, the recursively child columns are hidden, and the current column is shown.

The following screen shot shows the control's expandable header:



The following movie shows how you can use the Expandable Header support.

property Column.Expanded as Boolean

Expands or collapses the column.

Type	Description
Boolean	A Boolean expression that specifies whether the column is expanded or collapsed.

The Expanded property expands programmatically a column. The ExpandColumns property specifies the list of columns to be shown when the current column is expanded. The DisplayExpandButton property specifies whether the column's header displays a +/- (expanding button), to let user expands or collapse the column, when it is clicked. The control fires the ViewEndChanging(exLayoutChange) event when the user expands or collapse a column. Use the ExpandItem property to expand or collapse an item. The Index property indicates the column's index. The Visible property specifies whether a column is Visible or hidden.

property Column.Filter as String

Specifies the column's filter when the filter type is exFilter, exPattern, exDate, exNumeric, exCheck or exImage.

Type	Description
String	A string expression that specifies the column's filter.

- If the FilterType property is **exFilter** the Filter property indicates the list of values being included when filtering. The values are separated by '|' character. For instance if the Filter property is "CellA|CellB" the control includes only the items that have captions like: "CellA" or "CellB".

- If the FilterType is **exPattern** the Filter property defines the list of patterns used in filtering. The list of patterns is separated by the '|' character. A pattern filter may contain the wild card characters like '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character. The '|' character separates the options in the pattern. For instance: '1*|2*' specifies all items that start with '1' or '2'.

- If the FilterType property is **exDate**, the Filter property should be of "*[dateFrom] to [dateTo]*" format, and it indicates that only items between a specified range of dates will be included. If the dateFrom value is missing, the control includes only the items before the dateTo date, if the dateTo value is missing, the control includes the items after the dateFrom date. If both dates (dateFrom and dateTo) are present, the control includes the items between this interval of dates. For instance, the "2/13/2004 to" includes all items after 2/13/2004 inclusive, or "2/13/2004 to Feb 14 2005" includes all items between 2/13/2004 and 2/14/2004.

- If the FilterType property is **exNumeric**, the Filter property may include operators like <, <=, =, <>, >= or > and numbers to define rules to include numbers in the control's list. The Filter property should be of the following format "*operator number [operator number ...]*". For instance, the "> 10" indicates all numbers greater than 10. The "<>10 <> 20" filter indicates all numbers except 10 and 20. The "> 10 < 100" filter indicates all numbers greater than 10 and less than 100. The ">= 10 <= 100 <> 50" filter includes all numbers from 10 to 100 excepts 50. The "10" filter includes only 10 in the list. The "=10 =20" includes no items in the list because after control filters only 10 items, the second rule specifies only 20, and so we have no items. The Filter property may include unlimited rules. A rule is composed by an operator and a number. The rules are separated by space characters.

- If the FilterType property is **exCheck** the Filter property may include "0" for unchecked items, and "1" for checked items. The CellState property specifies the state of the

cell's checkbox. If the Filter property is empty, the filter is not applied to the column, when the ApplyFilter method is called. The drop down filter window displays the (All), (Checked) and (Unchecked) items.

- If the FilterType property is **exImage** the Filter property indicates the list of icons (index of the icon being displayed) being filtered. The values are separated by '|' character. The CellImage property indicates the index of the icon being displayed in the cell. For instance, the '1|2' indicates that the filter includes the cells that display first or the second icon (with the index 1 or 2). The drop down filter window displays the (All) item and the list of icons being displayed in the column.

The Filter property has no effect if the FilterType property is one of the followings: **exAll, exBlanks** and **exNonBlanks**

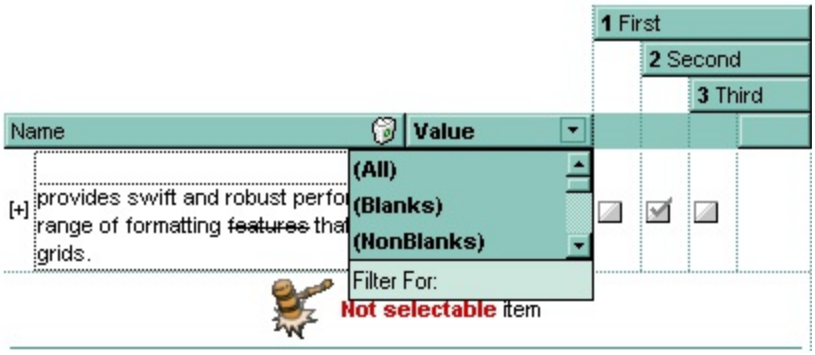
The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property. The ClearFilter method clears the Filter and the FilterType properties. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the CustomFilter property to define you custom filters. Use the ShowFilter method to show programmatically the column's drop down filter window.

property Column.FilterBarDropDownWidth as Double

Specifies the width of the drop down filter window proportionally with the width of the column.

Type	Description
Double	A double expression that indicates the width of the drop down filter window proportionally with the width of the column. If the FilterBarDropDownWidth expression is negative, the absolute value indicates the width of the drop down filter window in pixels. Else, the value indicates how many times the width of the column is multiply to get the width of the drop down filter window

By default, the FilterBarDropDownWidth property is 1, and so, the width of the drop down filter window coincides with the width of the column. Use the Width property to specify the width of the column. Use FilterBarDropDownHeight property to specify the height of the drop down filter window. Use the FilterBarHeight property to specify the height of the control's filter bar. Use the DisplayFilterButton property to display a filter button to the column's caption.



property Column.FilterList as FilterListEnum

Specifies whether the drop down filter list includes visible or all items.

Type	Description
FilterListEnum	A FilterListEnum expression that indicates the items being included in the drop down filter list.

By default, the FilterList property is exAllItems. Use the FilterList property to specify the items being included in the column's drop down filter list. Use the DisplayFilterButton property to display the column's filter bar button. The DisplayFilterDate property specifies whether the drop down filter window displays a date selector to specify the interval dates to filter for. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the exSortItemsAsc flag to sort ascending the values in the drop down filter list. For instance, the **exAllItems OR exSortItemsAsc** specifies that the drop down filter window lists all items in ascending order. Add the exIncludeInnerCells flag if you require adding the inner cells value to the drop down filter window.

property Column.FilterOnType as Boolean

Filters the column as user types characters in the drop down filter window.

Type	Description
Boolean	A Boolean expression that specifies whether the column gets filtered as the user types characters in the drop down filter window.

By default, the FilterOnType property is False. The Filter-On-Type feature allows you to filter the control's data based on the typed characters. Use the DisplayFilterButton property to add a drop down filter button to the column's header. The Filter-On-Type feature works like follows: User clicks the column's drop down filter button, so the drop down filter window is shown. Use starts type characters, and the control filters the column based on the typed characters as it includes all items that starts with typed characters, if the AutoSearch property is exStartWith, or include in the filter list only the items that contains the typed characters, if the AutoSearch property is exContains. Click the X button on the filterbar, and so the control removes the filter, and so all data is displayed. Once, the FilterOnType property is set on True, the column's FilterType property is changed to exPattern, and the the Filter property indicates the typed string. Use the FilterCriteria property to specify the expression being used to filter the control's data when multiple columns are implied in the filter. Use the FilterHeight property to specify the height of the control's filterbar that's displayed on the bottom side of the control, once a filter is applied. The "Filter For" (pattern) field in the drop down filter window is always shown if the FilterOnType property is True, no matter of the DisplayFilterPattern property.

property Column.FilterType as FilterTypeEnum

Specifies the column's filter type.

Type	Description
FilterTypeEnum	A FilterTypeEnum expression that indicates the filter's type.

The FilterType property defines the filter's type. By default, the FilterType is exAll. No filter is applied if the FilterType is exAll. The Filter property defines the column's filter. Use the DisplayFilterButton property to display the column's filter button. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the FilterCriteria property to filter items using the AND, OR and NOT operators.

The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property. The ClearFilter method clears the Filter and the FilterType properties.

If the FilterType property is exNumeric, the drop down filter window doesn't display the filter list that includes items "(All)", "(Blanks)", ... and so on.

property Column.FormatColumn as String

Specifies the format to display the cells in the column.

Type	Description
String	A string expression that defines the format to display the cell, including HTML formatting, if the cell supports it.

By default, the FormatColumn property is empty. The cells in the column use the provided format only if is valid (not empty, and syntactically correct), to display data in the column. The FormatColumn property provides a format to display all cells in the column using a predefined format. The expression may be a combination of variables, constants, strings, dates and operators, and **value**. The ***value*** operator gives the value to be formatted. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by # character, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM. The cell's HTML format is applied only if the CellValueFormat or Def(exCellCaptionFormat) is exHTML. If valid, the FormatColumn is applied to all cells for which the CellCaptionFormat property is not exComputedField. This way you can specify which cells use or not the FormatColumn property. The FormatCell property indicates the individually predefined format to be applied to particular cells. The FormatColumn and FormatCell properties support auto-numbering functions like explained bellow. The ComputedField property indicates the formula of the computed column.

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

For instance:

- the "*currency(value)*" displays the column using the current format for the currency ie, 1000 gets displayed as $1,000.00
- the "*longdate(date(value))*" converts the value to a date and gets the long format to display the date in the column, ie #1/1/2001# displays instead Monday, January 01, 2001
- the "*'' + ((0:=proper(value)) left 1) + '' + (=:0 mid 2)*" converts the name to proper, so the first letter is capitalized, bolds the first character, and let unchanged the rest, ie a "mihai filimon" gets displayed "**M**ihai Filimon".
- the "*len(value) ? ((0:=dbl(value)) < 10 ? '<fgcolor=808080><font ;7>' : '') + currency(=:0)*" displays the cells that contains not empty daya, the value in currency format, with a different font and color for values less than 10, and bolded for those that are greater than 10, as can see in the following screen shot in the column (A+B+C):

The **value** keyword in the FormatColumn property indicates the value being formatted.

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, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

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

The following **VB** sample shows how can I display the column using currency:

```
With View1
  .Columns.Add("Currency").FormatColumn = "currency(dbl(value))"
  With .Items
    .AddItem "1.23"
    .AddItem "2.34"
```

```
    .AddItem "0"
    .AddItem 5
    .AddItem "10000.99"
  End With
End With
```

The following **VB.NET** sample shows how can I display the column using currency:

```
With AxView1
  .Columns.Add("Currency").FormatColumn = "currency(dbl(value))"
  With .Items
    .AddItem "1.23"
    .AddItem "2.34"
    .AddItem "0"
    .AddItem 5
    .AddItem "10000.99"
  End With
End With
```

The following **C++** sample shows how can I display the column using currency:

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXG2ANTTLib' for the library: 'ExView 1.0 Control Library'

  #import "C:\\Windows\\System32\\ExView.dll"
  using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IViewPtr spView1 = GetDlgItem(IDC_G2ANTT1)->GetControlUnknown();
((EXG2ANTTLib::IColumnPtr)(spView1->GetColumns()->Add(L"Currency")))-
>PutFormatColumn(L"currency(dbl(value))");
EXG2ANTTLib::IItemsPtr var_Items = spView1->GetItems();
  var_Items->AddItem("1.23");
  var_Items->AddItem("2.34");
  var_Items->AddItem("0");
  var_Items->AddItem(long(5));
  var_Items->AddItem("10000.99");
```

The following **C#** sample shows how can I display the column using currency:

```
(axView1.Columns.Add("Currency") as EXG2ANTTLib.Column).FormatColumn =
"currency(dbl(value))";
EXG2ANTTLib.Items var_Items = axView1.Items;
   var_Items.AddItem("1.23");
   var_Items.AddItem("2.34");
   var_Items.AddItem("0");
   var_Items.AddItem(5);
   var_Items.AddItem("10000.99");
```

The following **VFP** sample shows how can I display the column using currency:

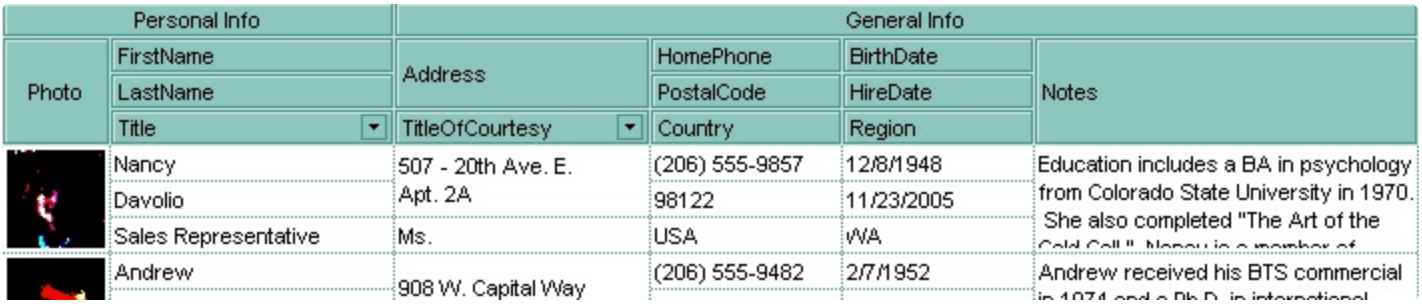
```
with thisform.View1
   .Columns.Add("Currency").FormatColumn = "currency(dbl(value))"
   with .Items
      .AddItem("1.23")
      .AddItem("2.34")
      .AddItem("0")
      .AddItem(5)
      .AddItem("10000.99")
   endwith
endwith
```

property Column.FormatLevel as String

Retrieves or sets a value that indicates the layout of columns being displayed in the column's header.

Type	Description
String	A string expression that indicates a CRD string that layouts the column's header. The Index elements in the CRD strings indicates the index of the column being displayed. The Caption elements in the CRD string support built-in HTML format.

By default, the FormatLevel property is empty. The FormatLevel property indicates the layout of the column in the control's header bar. Use the HeaderVisible property to show or hide the control's header bar. Use the HeaderHeight property to specify the height of the level in the control's header bar. Use the FormatLevel property to display multiple levels in the column's header. Use the LevelKey property to display neighbor columns on multiple levels. If the FormatLevel property is empty, the control displays the Caption or the HTMLCaption of the column. If the FormatLevel property is not empty it indicates the layout of the column being displayed. For instance, the FormatLevel = "1,2" indicates that the column's header is horizontally divided such as the left part displays the caption of the first column, and the right part displays the caption of the second column. Use the Visible property to specify whether a column is visible or hidden. Use the Add method to add new columns to the control. Use the DataSource property to bound the control to a recordset. Use the Def(exCellFormatLevel) property to specify the layout for all cells in the same column. Use the CellFormatLevel property to indicate the layout for a specific cell.



property Column.GroupByFormatCell as String

Indicates the format of the cell to be displayed when the column gets grouped by.

Type	Description
String	A String expression that may specify HTML format, **<caption>** and **value** keywords as explained bellow.

By default, the GroupByFormatCell property is "'<caption> (' + value + ')'", which indicates that the grouping label is shown in bold, followed by the computed value of the GroupByTotalField property. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted. The GroupByTotalField property determines the formula to be applied to the column when it gets grouped. *When the control is performing a group-by operation, the Items.FormatCell(Item,Items.GroupItem(Item)) property is set on GroupByFormatCell property, where the Item is the handle of the item being added during grouping or the Item parameter of the ViewItemUpdate(exAddGroupItem) event.*

In conclusion,

- *the **<caption>** keyword in the GroupByFormatCell property is replaced with the grouping label/value, and the result expression is passed to the FormatCell property.*
- *the **value** keyword indicates the computed value of the GroupByTotalField property.*

For instance:

- the *"'<caption> (' + currency(value) + ')'"* displays the grouping label, and the aggregate field as a currency, as specified in the regional settings.
- the *"'<caption> (' + currency(value) + `, inc. VAT ` + currency(1.19*value) + ')'"* displays the grouping label, and the aggregate field, including a computed field (VAT) as a currency, as specified in the regional settings.
- the *"'<caption> <fgcolor=808080>(Total ' + (value format ``) + ') </fgcolor>'"* displays the grouping label, and the aggregate field as a current number format, as specified in the regional settings, with a different font and foreground color.

The **value** keyword in the GroupByFormatCell property indicates the value to be formatted (as a result of the GroupByTotalField property):

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, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)"*

converts the value of the column with the index 1, to integer.

- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

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

property Column.GroupByTotalField as String

Indicates the aggregate formula to be displayed when the column gets grouped by.

Type	Description
String	A String expression that indicates the formula to be displayed on the grouping caption.

By default, the GroupByTotalField property is "count(current,rec,1)", which count recursively leaf items (implies recursively leaf items) of the grouping item. At runtime, the computed value of this formula is replaced in the HTML format being specified by the GroupByFormatCell property, for the **value** keyword. *When the control is performing a group-by operation, the Items.CellValue(Item,Items.GroupItem(Item)) property is set on GroupByTotalField property, and the Items.CellValueFormat(Item,Items.GroupItem(Item)) is exHTML + exTotalField (5), where the Item is the handle of the item being added during grouping or the Item parameter of the ViewItemUpdate(exAddGroupItem) event.* The GroupByTotalField property determines the formula to be applied to the column when it gets grouped. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted.

For instance

- "*count(current,dir,1)*" counts the number of child items (not implies recursively child items).
- "*count(current,all,1)*" counts the number of all child items (implies recursively child items).
- "*sum(parent,dir,%1=0?0:1)*" counts the not-zero values in the second column (%1)
- "*sum(parent,dir,%1 + %2)*" indicates the sum of all cells in the second (%1) and third (%2) column that are directly descendent from the parent item.
- "*sum(all,rec,%1 + %2)*" sums all leaf cells in the second (%1) and third (%2) columns.

The syntax for the GroupByTotalField property property should be:
aggregate(**list**,**direction**,**formula**) where:

aggregate must be one of the following:

- *sum* - calculates the sum of values.
- *min* - retrieves the minimum value.
- *max* - retrieves the maximum value.
- *count* - counts the number of items.
- *avg* - calculates the average of values.

list must be one of the following:

- a *long* expression that specifies the index of the item being referred.
- a predefined string expression as follows:
  - *all* - indicates all items, so the formula is being applied to all items. The direction has no effect.
  - *current* - the current item.
  - *parent* - the parent item.
  - *root* - the root item.

direction must be one of the following:

- *dir* - collects the direct descendents.
- *rec* - collects the leaf descendents (leaf items). A leaf item is an item with no child items.
- *all* - collects all descendents.

Currently, the following items are excluded by aggregate functions:

- *not-sortable items*. The SortableItem property specifies whether the item can be sorted (a sortable item can change its position after sorting, while a not-sortable item keeps its position after sorting.
- *not-selectable items*. The SelectableItem property specifies whether the user can selects/focus the specified item.
- *divider items*. The ItemDivider property specifies whether the item displays a single cell, instead displaying whole cells.

In conclusion, aggregate functions counts ONLY items that are:

- *sortable*, SortableItem is True, by default.
- *selectable*, SelectableItem is True, by default.
- *not divider*, ItemDivider is -1, by default.

Shortly, by setting to a different value to any of these properties, makes the item to be ignored by the aggregate functions.

For instance

- count(current,dir,1) counts the number of child items (not implies recursively child items).
- count(current,all,1) counts the number of all child items (implies recursively child items).
- count(current,rec,1) counts the number of leaf items (implies recursively leaf items).
- count(current,rec,1) counts the number of leaf items (a leaf item is an item with no child items).
- sum(parent,dir,%1=0?0:1) counts the not-zero values in the second column (%1)

- sum(parent,dir,%1 + %2) indicates the sum of all cells in the second (%1) and third (%2) column that are directly descendent from the parent item.
- sum(all,rec,%1 + %2) sums all leaf cells in the second (%1) and third (%2) columns.

property Column.HeaderAlignment as AlignmentEnum

Specifies the alignment of the column's caption.

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

Use the HeaderAlignment property to align the column's caption inside the column's header. Use the Alignment property to align the cells into a column. Use the HeaderImageAlignment property to align the column's icon inside the column's header. Use the CellHAlignment property to align a cell. The RightToLeft property automatically flips the order of the columns

property Column.HeaderBold as Boolean

Retrieves or sets a value that indicates whether the column's caption should appear in bold.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in bold.

The HeaderBold property specifies whether the column's caption should appear in bold. Use the CellBold or ItemBold properties to specify whether the cell or item should appear in bold. Use the HTMLCaption property to specify portions of the caption using different colors, fonts. Use the HeaderItalic, HeaderUnderline or HeaderStrikeOut property to specify different font attributes when displaying the column's caption.

property Column.HeaderImage as Long

Retrieves or sets a value indicating the index of an Image in the Images collection, which is displayed to the column's header.

Type	Description
Long	A long expression that indicates the index of icon in the Images collection, that's displayed on the column's header. 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 the HeaderImage property to add an icon to the column's header. The HeaderImage property does not set the icon for any of the column cells. Use the CellImage property to set an icon for a particular cell. Use the HeaderImageAlignment property to align the icon in the column's header. If the index of the icon in the column's header doesn't exist in the Images collection, no icon is displayed. Use the DisplaySortIcon property to specify whether the control displays the sorting icon when the user sorts a column. Use the Images method to assign a list of icons to the control at runtime. Use the <img> built-in HTML tag to insert multiple custom size picture/icons to the same header.

property Column.HeaderImageAlignment as AlignmentEnum

Retrieves or sets the alignment of the image in the column's header.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the icon in the column's header.

By default, the image is left aligned. Use the HeaderImageAlignment property to aligns the icon in the column's header. Use the HeaderImage property to attach an icon to the column's header. The RightToLeft property automatically flips the order of the columns

property Column.HeaderItalic as Boolean

Retrieves or sets a value that indicates whether the column's caption should appear in italic.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in italic.

Use the HeaderItalic property to specify whether the column's caption should appear in italic. Use the CellItalic or ItemItalic properties to specify whether the the cell or the item should appear in italic. Use the HeaderBold, HeaderUnderline or HeaderStrikeOut property to specify different font attributes when displaying the column's caption.

property Column.HeaderStrikeOut as Boolean

Retrieves or sets a value that indicates whether the column's caption should appear in strikeout.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in strikeout.

Use the HeaderStrikeOut property to specify whether the column's caption should appear in strikeout. Use the CellStrikeOut or ItemStrikeOut properties to specify whether the cell or the item should appear in strikeout. Use the HeaderItalic, HeaderUnderline or HeaderBold property to specify different font attributes when displaying the column's caption.

property Column.HeaderUnderline as Boolean

Retrieves or sets a value that indicates whether the column's caption should appear in underline.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in underline.

Use the HeaderUnderline property to specify whether the column's caption should appear in underline. Use the CellUnderline or ItemUnderline properties to specify whether the cell or the item should appear in underline. Use the HeaderItalic, HeaderBold or HeaderStrikeOut property to specify different font attributes when displaying the column's caption.

property Column.HeaderVertical as Boolean

Specifies whether the column's header is vertically displayed.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption is vertically printed.

Use the HeaderVertical property to display vertically the column's caption. Use the HeaderAlignment property to align the caption in the column's header. Use the Caption property to assign a caption to a column. Use the HTMLCaption property to specify an HTML caption to a column. Use the HeaderImage property to assign an icon to a column.

property Column.HTMLCaption as String

Retrieves or sets the text in HTML format displayed in the column's header.

Type	Description
String	A string expression that indicates the column's caption using built-in HTML tags.

If the HTMLCaption property is empty, the Caption property is displayed in the column's header. If the HTMLCaption property is not empty, the control uses it when displaying the column's header. Use the HeaderHeight property to change the height of the control's header bar. Use the HeaderVertical property to display vertically the column's caption. Use the HeaderAlignment property to align the caption in the column's header. Use the HeaderImage property to assign an icon to a column. The list of built-in HTML tags supported are here. Use the FormatLevel property to display multiple levels in the column's header.

property Column.Index as Long

Returns a value that represents the index of an object in a collection.

Type	Description
Long	A long expression that indicates the column's index.

The Index property of the Column is read only. Use the Position property to change the column's position. The Columns collection is zero based, so the Index property starts at 0. The last added column has the Index set to Columns.Count - 1. When a column is removed from the collection, the control updates all indexes. Use the Visible property to hide a column. Use the Columns property to access column from it's index.

property Column.Key as String

Retrieves or sets a the column's key.

Type	Description
String	A string expression that defines the column's key

The column's key defines a column when using the Item property. Use the Index or the Key property to identify a column, when using the Columns property.

property Column.LevelKey as Variant

Retrieves or sets a value that indicates the key of the column's level.

Type	Description
Variant	A Variant expression that indicates the key of the column's level.

By default, the LevelKey is empty. The control's header displays multiple levels if there are two or more neighbor columns with the same non empty level key. The HeaderHeight property specifies the height of one level when multiple levels header is on. Use the BackColorLevelHeader property to specify the control's level header area. The BackColorHeader property specifies the background color for column's captions. Use the FormatLevel property to display multiple levels in the column's header.

property Column.MaxWidthAutoResize as Long

Retrieves or sets a value that indicates the maximum column's width when the WidthAutoResize is True.

Type	Description
Long	A long expression that the maximum column's width when the WidthAutoResize is True.

If the WidthAutoResize property is False, the MaxWidthAutoResize and MinWidthAutoResize properties have no effect. The MaxWidthAutoResize property specifies the maximum column's width. The control recalculates the column's width each time when an item is expanded or collapsed. If the MaxWidthAutoResize property is -1, there is no maximum value for the column's width. Use the WidthAutoResize, MaxWidthAutoResize and MinWidthAutoResize properties when you don't want to have truncated the caption for cells in the column. Use the ColumnAutoResize property to specify whether the control resizes the visible columns so they fit the control's client area.

property Column.MinWidthAutoResize as Long

Retrieves or sets a value that indicates the minimum column width when the WidthAutoResize is True.

Type	Description
Long	A long expression that indicates the minimum column's width when the WidthAutoResize is True.

If the WidthAutoResize property is False, the MaxWidthAutoResize and MinWidthAutoResize properties have no effect. The MinWidthAutoResize property specifies the minimum column's width. The control recalculates the column's width each time when an item is expanded or collapsed. Use the WidthAutoResize, MaxWidthAutoResize and MinWidthAutoResize properties when you don't want to have truncated the caption for cells in the column. Use the ColumnAutoResize property to specify whether the control resizes the visible columns so they fit the control's client area.

property Column.PartialCheck as Boolean

Specifies whether the column supports partial check feature.

Type	Description
Boolean	A boolean expression that indicates whether the column supports partial check feature.

The PartialCheck property specifies that the column supports partial check feature. By default, the PartialCheck property is False. Use the CellHasCheckBox property to associate a check box to a cell. Use the Def property to assign a cell box for the entire column. Use the CellState property to determine the cell's state. If the PartialCheck property is True, the CellState property has three states: 0 - Unchecked, 1 - Checked and 2 - Partial Checked. The control supports partial check feature for any column that your control contains. Use the Add method to add new columns to the control. The control fires the ViewItemStateStartChanging(exCheckItem) / ViewItemStateEndChanging(exCheckItem) event when the user clicks a checkbox or a radio button in the control.

property Column.Position as Long

Retrieves or sets a value that indicates the position of the column in the header bar area.

Type	Description
Long	A long expression that indicates the position of the column in the header bar area

The column's index is not the same with the column's position. The Index property of Column cannot be changed by the user. Use the Position property to change the column's position. The EnsureVisibleColumn method ensures that a given column fits the control's client area. Use the Visible property to hide a column. Use the Width property to specify the column's width.

property Column.Selected as Boolean

Retrieves or sets a value that indicates whether the cell in the column is selected.

Type	Description
Boolean	A boolean expression that specifies whether the cell in the column is selected.

Use the Selected property to determine the cells being selected, when FullRowSelect property is exRectSel. Use the SelectItem property to programmatically selects an item. Use the SingleSel property to allow multiple items or cells in the selection. The control fires the ViewItemStateStartChanging(exActivateItem) / ViewItemStateEndChanging(exActivateItem) event when user changes the selection.

method Column.ShowFilter ([Options as Variant])

Shows the column's filter window.

Type	Description
Options as Variant	A string expression that indicates the position (in screen coordinates) and the size (in pixels) where the drop down filter window is shown. The Options parameter is composed like follows: • the first parameter indicates the X coordinate in screen coordinate, -1 if the current cursor position is used, or empty if the coordinate is ignored • the second parameter indicates the Y coordinate in screen coordinate, -1 if the current cursor position is used, or empty if the coordinate is ignored • the third parameter indicates the width in pixels of the drop down window, or empty if the width is ignored • the forth parameter indicates the height in pixels of the drop down window, or empty if the height is ignored By default, the drop down filter window is shown at its default position bellow the column's header.

Use the ShowFilter method to show the column's drop down filter programmatically. By default, the drop down filter window is shown only if the user clicks the filter button in the column's header, if the DisplayFilterButton property is True. The drop down filter window if the user selects a predefined filter, or enters a pattern to match. If the Options parameter is missing, or all parameters inside the Options are missing, the size of the drop down filter window is automatcially computed based on the FilterBarDropDownWidth property and FilterBarDropDownHeight property. Use the ColumnFromPoint property to get the index of the column from the point.

property Column.SortOrder as SortOrderEnum

Specifies the column's sort order.

Type	Description
SortOrderEnum	A SortOrderEnum expression that indicates the column's sort order.

The SortOrder property determines the column's sort order. By default, the SortOrder property is SortNone. Use the SortOrder property to sort a column at runtime. Use the SortType property to determine the way how the column is sorted. Use the AllowSort property to avoid sorting a column when the user clicks the column. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. If the control supports sorting by multiple columns, the SortOrder property adds or removes the column to the sorting columns collection. For instance, if the SortOrder property is set to SortAscending or SortDescending the column is added to the sorting columns collection. If the SortOrder property is set to SortNone the control removes the column from its sorting columns collection. The SortPosition property changes the position of the column in the control's sort bar. Use the DefaultSortOrder property to specify the column's default sort order, when the user first clicks the column's header. The EnsureOnSort property prevents scrolling the control's content when the user sorts items. The SortableItem property specifies whether the item keeps its position after sorting.

The control automatically sorts a column when the user clicks the column's header, if the SortOnClick property is exDefaultSort. If the SortOnClick property is exNoSort, the control disables sorting the items when the user clicks the column's header. There are two methods to get the items sorted like follows:

- Using the SortOrder property of the Column object::

  ```
  View1.Columns(ColIndex).SortOrder = SortAscending
  ```

  The SortOrder property adds the sorting icon to the column's header, if the DisplaySortIcon property is True.

- Using the SortChildren method of the Items collection. The SortChildren sorts the items. The SortChildren method sorts the child items of the given parent item in the control. SortChildren will not recourse through the tree, only the immediate children of the item will be sorted. The following sample sort descending the list of root items on the "Column 1"(if your control displays a list, all items are considered being root items

```
View1.Items.SortChildren 0, "Column 1", False
```

The SortType property of the Column object specifies the way how a column gets sorted.

By default, a column gets sorted as string. If you need to sort your dates, the following snippet of code should be used:

```
With View1
      With .Columns(0)
         .SortType = SortDate
      End With
End With
```

If you need to sort a column using your special way you may want to use the SortType = SortUserData, or SortType = SortCellData that sorts the column using CellData / CellSortData property for each cell in the column. In this case, the CellData or CellSortData property holds numeric values only.

property Column.SortPosition as Long

Returns or sets a value that indicates the position of the column in the sorting columns collection.

Type	Description
Long	A long expression that indicates the position of the column in the control's sort bar. The collection is 0 - based.

Use the SortPosition to change programmatically the position of the column in the control's sort bar. Use the SingleSort property to allow sorting by multiple columns. Use the SortBarVisible property to show the control's sort bar. Use the SortOrder property to add columns to the control's sort bar. Use the ItemBySortPosition property to get the columns being sorted in their order. Use the AllowSort property to avoid sorting a column when the user clicks the column.

property Column.SortType as SortTypeEnum

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

Type	Description
SortTypeEnum	A SortTypeEnum expression that indicates the way how control sorts the column.

By default, the column's sort type is string. Use the SortType property to specify how the control sorts the column. Use the DisplaySortIcon property to hide the sort icon displayed when the column was sorted. Use the SortChildren method to sort items. Use the CellCaption property to get the string being displayed in the cell. Use the CellValue property to specify the cell's value. Use the CellSortData to specify the data being sorted when the SortType property is SortCellData or SortCellDataString. Use the CellData property to specify the values being sorted if the SortType property is SortUserData. The SortPosition property changes the position of the column in the sorting columns collection. the SingleSort property to specify whether the control supports sorting by single or multiple columns. The SortOrder property determines the column's sort order. The SortableItem property specifies whether the item keeps its position after sorting.

property Column.ToolTip as String

Specifes the column's tooltip description.

Type	Description
String	A string expression that defines the column's tooltip. The column's tooltip supports built-in HTML format.

By default, the ToolTip property is "..." (three dots). Use the ToolTip property to assign a tooltip to a column. If the ToolTip property is "...", the control displays the column's caption if it doesn't fit the column's header. Use the Caption or HTMLCaption property to specify the caption of the column. The column's tooltip shows up when the cursor hovers the column's header. Use the CellToolTip property to assign a tooltip to a cell. The ToolTipWidth property specifies a value that indicates the width of the tooltip window, in pixels.

property Column.Visible as Boolean

Retrieves or sets a value indicating whether the column is visible or hidden.

Type	Description
Boolean	A boolean expression indicating whether the column is visible or hidden.

Use the Visible property to hide a column. Use the Width property to resize the column. The ColumnAutoResize property specifies whether the visible columns fit the control's client area. Use the Position property to specify the column's position. Use the HeaderVisible property to show or hide the control's header bar. Use the ColumnFromPoint property to get the column from point. Use the Remove method to remove a column. Use the FormatLevel property to display multiple levels in the column's header.

property Column.Width as Long

Retrieves or sets the column's width.

Type	Description
Long	A long expression that indicates the column's width.

The Width property resizes a column at runtime. Use the AutoWidth property to compute the width that's required to fit all cells in the column. Use the WidthAutoResize property to automatically resize the column while the user expands or collapses items. Use the Visible property to hide a column. The ColumnAutoResize property specifies whether the visible columns fit the control's client area. If the ColumnAutoResize property is True, the Width property may not resize the column to the desired value, because all visible columns must fit the control's client area. By default, the control adds horizontal scroll bar when required. Use the ScrollBars property to add or remove the control's scroll bars. Use the Visible property to hide the column. Use the FilterBarDropDownWidth property to specify the width of the drop down filter window.

property Column.WidthAutoResize as Boolean

Retrieves or sets a value that indicates whether the column is automatically resized according to the width of the contents within the column.

Type	Description
Boolean	A boolean expression that indicates whether the column is automatically resized according to the width of the contents within the column.

Use the WidthAutoResize property if you need to display the entire caption of each cell in the column. If the WidthAutoResize property is True, the user is not able to resize the column, so the AllowSizing property has no effect in this case. Use the ColumnAutoResize property to specify whether the control resizes the visible columns so they fit the control's client area. You can use the AutoWidth property to computes the column's width to fit its content. For instance, if you have a tree with one column, and this property True, you can simulate a simple tree, because the control will automatically add a horizontal scroll bar when required. Use the HeaderVisible property to hide the control's header bar. Use the BeginUpdate and EndUpdate method to maintain performance while adding columns and items to the control. Use the MinWidthAutoResize property to specify the minimum column width, while the WidthAutoResize property is True.

Columns object

The Columns object holds a collection of Column objects. The Columns collection supports the following properties and methods:

Name	Description
Add	Adds a Column object to the collection 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 Column of the Columns collection.
ItemBySortPosition	Returns a Column object giving its sorting position.
Remove	Removes a specific member from the Columns collection.
SortBarColumn	Returns the Column from control's SortBar giving its position.
SortBarColumnsCount	Retrieves the count of Columns, in the control's SortBar

method Columns.Add (ColumnCaption as String)

Adds a Column object to the collection and returns a reference to the newly created object.

Type	Description
ColumnCaption as String	A string expression that defines the column's caption

Return	Description
Variant	A Column object that represents the newly created column.

By default, the control has no columns. Use Add method to add new columns to the control. If the control contains no columns, you cannot add new items to the control. Use the Remove method to remove a specific column. If the control's DataSource property points to an ADO recordset the user doesn't need to add columns to the control. Instead, the Add method can be used to add computed fields for instance. Use the AddItem, InsertItem, DataSource properties to add new items to the control. Use the BeginUpdate and EndUpdate methods to prevent control from painting while adding columns or items. Use the Def property to specify default setting for cells in the column. Use the FormatLevel property to display multiple levels in the column's header

method Columns.Clear ()

Removes all objects in a collection.

Type	Description

Use the Clear method to remove all columns in the Columns collection. If the Clear method is called, the control removes also all items. Use the Remove method to Remove a particular column. Use the RemoveAllItems method to remove all items in the control.

property Columns.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	Counts the columns in the collection.

The Count property counts the columns in the collection. Use the Columns property to access the control's Columns collection. Use the Item property to access a column by its index or key. Use the Add method to add new columns to the control. Use the Remove method to remove a column. Use the Clear method to clear the columns collection.

property Columns.Item (Index as Variant) as Column

Returns a specific Column of the Columns collection.

Type	Description
Index as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or column's key.
Column	A Column object being accessed.

Use the Item property to access to a specific column. The Count property counts the columns in the control. Use the Columns property to access the control's Columns collection. The SortBarColumn / SortBarColumnsCount properties can be used to enumerate the columns in the control's sort bar. The Visible property indicates whether the column is visible or hidden. The Position property specifies the position of the column. The user can change the column's position by drag and drop, so the position of the column can be changed at runtime. Instead the Index property is a read only property that gives the index of the column in the collection.

property Columns.ItemBySortPosition (Position as Variant) as Column

Returns a Column object giving its sorting position.

Type	Description
Position as Variant	A long expression that indicates the position of column being requested.
Column	A Column object being accessed.

Use the ItemBySortPosition property to get the list of sorted columns in their order. Use the SortPosition property to specify the position of the column in the sorting columns collection. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. Use the SortOrder property to sort a column programmatically. The SortBarColumn / SortBarColumnsCount properties can be used to enumerate the columns in the control's sort bar.

method Columns.Remove (Index as Variant)

Removes a specific member from the Columns collection.

Type	Description
Index as Variant	A long expression that indicates the column's index being removed, or a string expression that indicates the column's caption or column's key

The Remove method removes a specific column in the Columns collection. Use Clear method to remove all Column objects. Use the Visible property to hide a column.

property Columns.SortBarColumn (Position as Variant) as Column

Returns the Column from control's SortBar giving its position.

Type	Description
Position as Variant	A long expression that specifies the position where the column is requested
Column	A Column object that specifies the sorted/grouped column at giving position, or empty if no column is found.

The SortBarColumn / SortBarColumnsCount properties can be used to enumerate the columns in the control's sort bar. Use the SortOrder property of the Column object on SortAscending / SortDescending to add a column to the sort bar, on SortNone to remove the column from the control's sort bar. Use the SortType property to determine the way how the column is sorted. Use the AllowSort property to avoid sorting a column when the user clicks the column. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. For instance, the SortBarColumnsCount counts the number of grouped columns, if the control's AllowGroupBy property is True.

property Columns.SortBarColumnsCount as Long

Retrieves the count of Columns, in the control's SortBar

Type	Description
Long	A long expression that specifies the number of columns being shown in the control's sort bar.

By default, the SortBarColumnsCount property is 0. The SortBarColumnsCount property counts the columns being shown in the sort bar. The SortBarColumn / SortBarColumnsCount properties can be used to enumerate the columns in the control's sort bar. Use the SortOrder property of the Column object on SortAscending / SortDescending to add a column to the sort bar, on SortNone to remove the column from the control's sort bar. Use the SortType property to determine the way how the column is sorted. Use the AllowSort property to avoid sorting a column when the user clicks the column. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. For instance, the SortBarColumnsCount counts the number of grouped columns, if the control's AllowGroupBy property is True.

ConditionalFormat object

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. Use the Add method to add new ConditionalFormat objects. Use the Item property to access a ConditionalFormat object. The ConditionalFormat object supports the following properties and method:

Name	Description
ApplyTo	Specifies whether the format is applied to items or columns.
BackColor	Retrieves or sets the background color for objects that match the condition.
Bold	Bolds the objects that match the condition.
ClearBackColor	Clears the background color.
ClearForeColor	Clears the foreground color.
Enabled	Specifies whether the condition is enabled or disabled.
Expression	Indicates the expression being used in the conditional format.
Font	Retrieves or sets the font for objects that match the criteria.
ForeColor	Retrieves or sets the foreground color for objects that match the condition.
Italic	Specifies whether the objects that match the condition should appear in italic.
Key	Checks whether the expression is syntactically correct.
StrikeOut	Specifies whether the objects that match the condition should appear in strikeout.
Underline	Underlines the objects that match the condition.
Valid	Checks whether the expression is syntactically correct.

property ConditionalFormat.ApplyTo as FormatApplyToEnum

Specifies whether the format is applied to items or columns.

Type	Description
FormatApplyToEnum	A FormatApplyToEnum expression that indicates whether the format is applied to items or to columns. If the ApplyTo property is less than zero, the format is applied to the items.

By default, the format is applied to items. The ApplyTo property specifies whether the format is applied to the items or to the columns. If the ApplyTo property is greater or equal than zero the format is applied to the column with the index ApplyTo. For instance, if the ApplyTo property is 0, the format is applied to the cells in the first column. If the ApplyTo property is 1, the format is applied to the cells in the second column, if the ApplyTo property is 2, the format is applied to the cells in the third column, and so on. If the ApplyTo property is -1, the format is applied to items.

The following VB sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With View1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_grid.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxView1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXCASCADETREELib.ConditionalFormat cf =
axView1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Bold = true;
cf.ApplyTo = (EXCASCADETREELib.FormatApplyToEnum)1;
```

The following VFP sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.View1.ConditionalFormats.Add("%1+%2<%0")
   .Bold = .t.
   .ApplyTo = 1
endwith
```

property ConditionalFormat.BackColor as Color

Retrieves or sets the background color for objects that match the condition.

Type	Description
Color	A color expression that indicates the background color for the object that match the criteria. 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 BackColor property to change the background color for items or cells in the column when a certain condition is met. Use the ForeColor property to specify the foreground color for objects that match the criteria. Use the ClearBackColor method to remove the background color being set using previously the BackColor property. If the BackColor property is not set, it retrieves 0. The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

property ConditionalFormat.Bold as Boolean

Bolds the objects that match the condition.

Type	Description
Boolean	A boolean expression that indicates whether the objects should appear in bold.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column. The following VB sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With View1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_grid.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxView1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXCASCADETREELib.ConditionalFormat cf =
axView1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Bold = true;
cf.ApplyTo = (EXCASCADETREELib.FormatApplyToEnum)1;
```

The following VFP sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.View1.ConditionalFormats.Add("%1+%2<%0")
    .Bold = .t.
    .ApplyTo = 1
endwith
```

method ConditionalFormat.ClearBackColor ()

Clears the background color.

Type	Description

Use the ClearBackColor method to remove the background color being set using previously the BackColor property. If the BackColor property is not set, it retrieves 0.

method ConditionalFormat.ClearForeColor ()

Clears the foreground color.

Type	Description

Use the ClearBackColor method to remove the foreground color being set using previously the ForeColor property. If the ForeColor property is not set, it retrieves 0.

property ConditionalFormat.Enabled as Boolean

Specifies whether the condition is enabled or disabled.

Type	Description
Boolean	A boolean expression that indicates whether the expression is enabled or disabled.

By default, all expressions are enabled. A format is applied only if the expression is valid and enabled. Use the Expression property to specify the format's formula. The Valid property checks whether the formula is valid or not valid. Use the Enabled property to disable applying the format for the moment. Use the Remove method to remove an expression from ConditionalFormats collection.

property ConditionalFormat.Expression as String

Indicates the expression being used in the conditional format.

Type	Description
String	A formal expression that indicates the formula being used in formatting. For instance, "%0+%1>%2", highlights the cells or the items, when the sum between first two columns is greater than the value in the third column

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. The Expression property specifies a formula that indicates the criteria to format the items or the columns. Use the ApplyTo property to specify when the items or the columns are formatted. Use the Add method to specify the expression at adding time. The Expression property may include variables, constants, operators or ( ) parenthesis. A variable is defined as %n, where n is the index of the column (zero based). For instance, the %0 indicates the first column, the %1, indicates the second column, and so on. A constant is a float expression (for instance, 23.45). Use the Valid property checks whether the expression is syntactically correct, and can be evaluated. If the expression contains a variable that is not known, 0 value is used instead. For instance, if your control has 2 columns, and the expression looks like "%2 +%1 ", the %2 does not exist, 0 is used instead. When the control contains two columns the known variables are %0 and %1.

The expression may be a combination of variables, constants, strings, dates and operators. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by two # characters, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM.

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, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*

- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

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

Usage examples:

1. "**1**", highlights all cells or items. Use this form, when you need to highlight all cells or items in the column or control.
2. "**%0 >= 0**", highlights the cells or items, when the cells in the first column have the value greater or equal with zero
3. "**%0 = 1 and %1 = 0**", highlights the cells or items, when the cells in the first column have the value equal with 0, and the cells in the second column have the value equal with 0
4. "**%0+%1>%2**", highlights the cells or the items, when the sum between first two columns is greater than the value in the third column
5. "**%0+%1 > %2+%3**", highlights the cells or items, when the sum between first two columns is greater than the sum between third and forth column.
6. "**%0+%1 >= 0 and (%2+%3)/2 < %4-5**", highlights the cells or the items, when the sum between first two columns is greater than 0 and the half of the sum between third and forth columns is less than fifth column minus 5.
7. "**%0 startwith 'A'**" specifies the cells that starts with A
8. "**%0 endwith 'Bc'**" specifies the cells that ends with Bc
9. "**%0 contains 'aBc'**" specifies the cells that contains the aBc string
10. "**lower(%0) contains 'abc'**" specifies the cells that contains the abc, AbC, ABC, and so on
11. "**upper(%0)**" retrieves the uppercase string
12. "**len(%0)>0**" specifies the not blanks cells
13. "**len %0 = 0**" specifies the blanks cells

The conditional format feature may change the cells and items as follows:

- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.

- Underline property. Underlines the cells or items
- Font property. Changes the font for cells or items.
- BackColor property. Changes the background color for cells or items, supports skins as well.
- ForeColor property. Changes the foreground color for cells or items.

The following VB samples bolds all items when the sum between first two columns is greater than 0:

```
View1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following C++ sample bolds all items when the sum between first two columns is greater than 0:

```
COleVariant vtEmpty;
m_grid.GetConditionalFormats().Add( "%0+%1>0", vtEmpty ).SetBold( TRUE );
```

The following VB.NET sample bolds all items when the sum between first two columns is greater than 0:

```
AxView1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following C# sample bolds all items when the sum between first two columns is greater than 0:

```
axView1.ConditionalFormats.Add("%0+%1>0", null).Bold = true
```

The following VFP sample bolds all items when the sum between first two columns is greater than 0:

```
thisform.View1.ConditionalFormats.Add("%0+%1>0").Bold = .t.
```

property ConditionalFormat.Font as IFontDisp

Retrieves or sets the font for objects that match the criteria.

Type	Description
IFontDisp	A Font object that's applied to items or columns.

Use the Font property to change the font for items or columns that match the criteria. Use the Font property only, if you need to change to a different font.

You can change directly the font attributes, like follows:

- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.
- Underline property. Underlines the cells or items

The following VB sample changes the font for ALL cells in the first column:

```
With View1.ConditionalFormats.Add("1")
   .ApplyTo = 0
   Set .Font = New StdFont
   With .Font
      .Name = "Comic Sans MS"
   End With
End With
```

property ConditionalFormat.ForeColor as Color

Retrieves or sets the foreground color for objects that match the condition.

Type	Description
Color	A color expression that indicates the foreground color for the object that match the criteria.

Use the ForeColor property to specify the foreground color for objects that match the criteria. Use the BackColor property to change the background color for items or cells in the column when a certain condition is met. Use the ClearForeColor method to remove the foreground color being set using previously the ForeColor property. If the ForeColor property is not set, it retrieves 0. The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

property ConditionalFormat.Italic as Boolean

Specifies whether the objects that match the condition should appear in italic.

Type	Description
Boolean	A boolean expression that indicates whether the objects should look in italic.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column. The following VB sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With View1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Italic = True
End With
```

The following C++ sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_grid.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetItalic( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxView1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Italic = True
End With
```

The following C# sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXCASCADETREELib.ConditionalFormat cf =
axView1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Italic = true;
cf.ApplyTo = (EXCASCADETREELib.FormatApplyToEnum)1;
```

The following VFP sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.View1.ConditionalFormats.Add("%1+%2<%0")
    .Italic = .t.
    .ApplyTo = 1
endwith
```

property ConditionalFormat.Key as Variant

Checks whether the expression is syntactically correct.

Type	Description
Variant	A String expression that indicates the key of the element

The Key property indicates the key of the element. Use the Add method to specify a key at adding time. Use the Remove method to remove a formula giving its key.

property ConditionalFormat.StrikeOut as Boolean

Specifies whether the objects that match the condition should appear in strikeout.

Type	Description
Boolean	A Boolean expression that indicates whether the objects should appear in strikeout.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column. The following VB sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With View1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_grid.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxView1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXCASCADETREELib.ConditionalFormat cf =
```

```
axView1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Bold = true;
cf.ApplyTo = (EXCASCADETREELib.FormatApplyToEnum)1;
```

The following VFP sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.View1.ConditionalFormats.Add("%1+%2<%0")
   .Bold = .t.
   .ApplyTo = 1
endwith
```

property ConditionalFormat.Underline as Boolean

Underlines the objects that match the condition.

Type	Description
Boolean	A boolean expression that indicates whether the objects are underlined.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column. The following VB sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With View1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Underline = True
End With
```

The following C++ sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_grid.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetUnderline( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxView1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Underline = True
End With
```

The following C# sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXCASCADETREELib.ConditionalFormat cf =
axView1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Underline = true;
cf.ApplyTo = (EXCASCADETREELib.FormatApplyToEnum)1;
```

The following VFP sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.View1.ConditionalFormats.Add("%1+%2<%0")
   .Underline = .t.
   .ApplyTo = 1
endwith
```

property ConditionalFormat.Valid as Boolean

Checks whether the expression is syntactically correct.

Type	Description
Boolean	A boolean expression that indicates whether the Expression property is valid.

Use the Valid property to check whether the Expression formula is valid. The conditional format is not applied to objects if expression is not valid, or the Enabled property is false. An empty expression is not valid. Use the Enabled property to disable applying the format to columns or items. Use the Remove method to remove an expression from ConditionalFormats collection.

ConditionalFormats object

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. The ConditionalFormats collection holds a collection of ConditionalFormat objects. Use the ConditionalFormats property to access the control's ConditionalFormats collection .The ConditionalFormats collection supports the following properties and methods:

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

method ConditionalFormats.Add (Expression as String, [Key as Variant])

Adds a new expression to the collection and returns a reference to the newly created object.

Type	Description
Expression as String	A formal expression that indicates the formula being used when the format is applied. Please check the Expression property that shows the syntax of the expression that may be used. For instance, the **"%0 >= 10 and %1 > 67.23"** means all cells in the first column with the value less or equal than 10, and all cells in the second column with a value greater than 67.23
Key as Variant	A string or long expression that indicates the key of the expression being added. If the Key parameter is missing, by default, the current index in the ConditionalFormats collection is used.

Return	Description
ConditionalFormat	A ConditionalFormat object that indicates the newly format being added.

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. Use the Add method to format cells or items based on values. Use the Add method to add new ConditionalFormat objects to the ConditionalFormats collection. By default, the ConditionalFormats collection is empty. A ConditionalFormat object indicates a formula and a format to apply to cells or items. The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column. Use the Expression property to retrieve or set the formula. Use the Key property to retrieve the key of the object. Use the Refresh method to update the changes on the control's content.

The conditional format feature may change the cells and items as follows:

- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.
- Underline property. Underlines the cells or items
- Font property. Changes the font for cells or items.
- BackColor property. Changes the background color for cells or items, supports skins as well.
- ForeColor property. Changes the foreground color for cells or items.

The following VB sample bolds all items when the sum between first two columns is greater than 0:

```
View1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following VB sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With View1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample bolds all items when the sum between first two columns is greater than 0:

```
COleVariant vtEmpty;
m_grid.GetConditionalFormats().Add( "%0+%1>0", vtEmpty ).SetBold( TRUE );
```

The following C++ sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_grid.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample bolds all items when the sum between first two columns is greater than 0:

```
AxView1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following VB.NET sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxView1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample bolds all items when the sum between first two columns is greater

than 0:

```
axView1.ConditionalFormats.Add("%0+%1>0", null).Bold = true
```

The following C# sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXCASCADETREELib.ConditionalFormat cf =
axView1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Bold = true;
cf.ApplyTo = (EXCASCADETREELib.FormatApplyToEnum)1;
```

The following VFP sample bolds all items when the sum between first two columns is greater than 0:

```
thisform.View1.ConditionalFormats.Add("%0+%1>0").Bold = .t.
```

The following VFP sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.View1.ConditionalFormats.Add("%1+%2<%0")
   .Bold = .t.
   .ApplyTo = 1
endwith
```

method ConditionalFormats.Clear ()

Removes all expressions in a collection.

Type	Description

Use the Clear method to remove all objects in the collection. Use the Remove method to remove a particular object from the collection. Use the Enabled property to disable a conditional format.

property ConditionalFormats.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A long expression that counts the number of elements in the collection.

Use the Item and Count property to enumerate the elements in the collection. Use the Expression property to get the expression of the format.

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As ConditionalFormat
For Each c In View1.ConditionalFormats
   Debug.Print c.Expression
Next
```

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With View1.ConditionalFormats
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Expression
   Next
End With
```

The following C++ sample enumerates all elements in the ConditionalFormats collection:

```
for ( long i = 0; i < m_grid.GetConditionalFormats().GetCount(); i++ )
{
   CConditionalFormat cf = m_grid.GetConditionalFormats().GetItem( COleVariant( i ) );
   OutputDebugString( cf.GetExpression() );
}
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As EXCASCADETREELib.ConditionalFormat
For Each c In AxView1.ConditionalFormats
   System.Diagnostics.Debug.Write(c.Expression)
Next
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With AxView1.ConditionalFormats
   For i = 0 To .Count - 1
      System.Diagnostics.Debug.Write(.Item(i).Expression)
   Next
End With
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
foreach (EXCASCADETREELib.ConditionalFormat c in axView1.ConditionalFormats)
   System.Diagnostics.Debug.Write(c.Expression);
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
for (int i = 0; i < axView1.ConditionalFormats.Count; i++)
   System.Diagnostics.Debug.Write(axView1.ConditionalFormats[i].Expression);
```

The following VFP sample enumerates all elements in the ConditionalFormats collection:

```
with thisform.View1.ConditionalFormats
   for i = 0 to .Count - 1
      wait .Item(i).Expression
   next
endwith
```

property ConditionalFormats.Item (Key as Variant) as ConditionalFormat

Returns a specific expression.

Type	Description
Key as Variant	A long expression that indicates the index of the element being accessed, or a string expression that indicates the key of the element being accessed.
ConditionalFormat	A ConditionalFormat object being returned.

Use the Item and Count property to enumerate the elements in the collection. Use the Expression property to get the expression of the format. Use the Key property to get the key of the format.

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As ConditionalFormat
For Each c In View1.ConditionalFormats
   Debug.Print c.Expression
Next
```

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With View1.ConditionalFormats
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Expression
   Next
End With
```

The following C++ sample enumerates all elements in the ConditionalFormats collection:

```
for ( long i = 0; i < m_grid.GetConditionalFormats().GetCount(); i++ )
{
   CConditionalFormat cf = m_grid.GetConditionalFormats().GetItem( COleVariant( i ) );
   OutputDebugString( cf.GetExpression() );
}
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As EXCASCADETREELib.ConditionalFormat
```

```
For Each c In AxView1.ConditionalFormats
   System.Diagnostics.Debug.Write(c.Expression)
Next
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With AxView1.ConditionalFormats
   For i = 0 To .Count - 1
      System.Diagnostics.Debug.Write(.Item(i).Expression)
   Next
End With
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
foreach (EXCASCADETREELib.ConditionalFormat c in axView1.ConditionalFormats)
   System.Diagnostics.Debug.Write(c.Expression);
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
for (int i = 0; i < axView1.ConditionalFormats.Count; i++)
   System.Diagnostics.Debug.Write(axView1.ConditionalFormats[i].Expression);
```

The following VFP sample enumerates all elements in the ConditionalFormats collection:

```
with thisform.View1.ConditionalFormats
   for i = 0 to .Count - 1
      wait .Item(i).Expression
   next
endwith
```

method ConditionalFormats.Remove (Key as Variant)

Removes a specific member from the collection.

Type	Description
Key as Variant	A Long or String expression that indicates the key of the element to be removed.

Use the Remove method to remove a particular object from the collection. Use the Enabled property to disable a conditional format. Use the Clear method to remove all objects in the collection.

Items object

The Items object contains a collection of items. Each item is identified by a handle HITEM. The HITEM is of long type. Each item contains a collection of cells. The number of cells is determined by the number of Column objects in the control. To access the Items collection use Items property of the control. Using the Items collection you can add, remove or change the control items. The Items collection can be organized as a hierarchy or as a tabular data. The Items collection supports the following properties and methods:

Name	Description
AcceptSetParent	Verifies whether the item can be child of another item.
AddItem	Adds a new item, and returns a handle to the newly created item.
CellBackColor	Retrieves or sets the cell's background color.
CellBold	Retrieves or sets a value that specifies whether the cell should appear in bold.
CellButtonAutoWidth	Retrieves or sets a value indicating whether the cell's button fits the cell's caption.
CellCaption	Gets the cell's display value.
CellChecked	Retrieves the cell's handle that is checked giving the radio group identifier.
CellData	Specifies the cell's extra data.
CellEnabled	Returns or sets a value that determines whether a cell can respond to user-generated events.
CellFont	Retrieves or sets the cell's font.
CellForeColor	Retrieves or sets the cell's foreground color.
CellFormatLevel	Specifies the arrangement of the fields inside the cell.
CellHAlignment	Retrieves or sets a value that indicates the alignment of the cell's caption.
CellHasButton	Retrieves or sets a value indicating whether the cell has associated a push button.
CellHasCheckBox	Retrieves or sets a value indicating whether the cell has associated a checkbox.
CellHasRadioButton	Retrieves or sets a value indicating whether the cell has associated a radio button.
CellHyperLink	Specifies whether the cell's is highlighted when the cursor mouse is over the cell.

CellImage	Retrieves or sets a value that indicates the index of icon in the cell.
CellImages	Specifies an additional list of icons shown in the cell.
CellItalic	Retrieves or sets a value that specifies whether the cell should appear in italic.
CellItem	Retrieves the handle of item that is the owner of a specfic cell.
CellMerge	Retrieves or sets a value that indicates the index of the cell that's merged to.
CellParent	Retrieves the parent of an inner cell.
CellPicture	Retrieves or sets the cell's picture.
CellPictureHeight	Retrieves or sets a value that indicates the height of the cell's picture.
CellPictureWidth	Retrieves or sets a value that indicates the width of the cell's picture.
CellRadioGroup	Retrieves or sets a value indicating the radio group where the cell is contained.
CellSingleLine	Retrieves or sets a value indicating whether the cell's caption is painted using one or more lines.
CellSortData	Specifies the cell's sort data.
CellState	Retrieves or sets the cell's state. Has effect only for check and radio cells.
CellStrikeOut	Retrieves or sets a value that specifies whether the cell should appear in strikeout.
CellToolTip	Retrieves or sets a value that indicates the cell's too tip
CellUnderline	Retrieves or sets a value that specifies whether the cell should appear in underline.
CellVAlignment	Retrieves or sets a value that indicates how the cell's caption is vertically aligned.
CellValue	Specifies the cell's value.
CellValueFormat	Specifies how the cell's caption is displayed.
CellWidth	Retrieves or sets a value that indicates the width of the inner cell.
ChildCount	Retrieves the number of children items.
ClearCellBackColor	Clears the cell's background color.

ClearCellForeColor	Clears the cell's foreground color.
ClearCellHAlignment	Clears the cell's alignment.
ClearItemBackColor	Clears the item's background color.
ClearItemForeColor	Clears the item's foreground color.
ComputeValue	Computes the value of a specified formula.
DefaultItem	Retrieves or sets a value that indicates the handle of the item used by Items properties in VFP.
EnableItem	Returns or sets a value that determines whether a item can respond to user-generated events.
EnsureVisibleItem	Ensures the given item is in the visible client area.
ExpandItem	Expands, or collapses, the child items of the specified item.
FindItem	Finds an item, looking for Value in ColIndex colum. The searching starts at StartIndex item.
FindItemData	Finds the item giving its data.
FindPath	Finds the item, given its path. The control searches the path on the SearchColumnIndex column.
FirstVisibleItem	Retrieves the handle of the first visible item in the control.
FocusItem	Retrieves the handle of item that has the focus.
FormatCell	Specifies the custom format to display the cell's content.
FullPath	Returns the fully qualified path of the referenced item in the ExGrid control. The value is taken from the column SearchColumnIndex.
GroupItem	Indicates a group item if positive, and the value specifies the index of the column that has been grouped.
InnerCell	Retrieves the inner cell.
InsertControlItem	Inserts a new item of ActiveX type, and returns a handle to the newly created item.
InsertItem	Inserts a new item, and returns a handle to the newly created item.
InsertObjectItem	Inserts a new item that hosts the giving object, and returns a handle to the newly created item.
IsItemLocked	Returns a value that indicates whether the item is locked or unlocked.
IsItemVisible	Checks if the specific item is in the visible client area.

ItemAllowSizing	Retrieves or sets a value that indicates whether a user can resize the item at run-time.
ItemAppearance	Specifies the item's appearance when the item hosts an ActiveX control.
ItemBackColor	Retrieves or sets a background color for a specific item.
ItemBold	Retrieves or sets a value that indicates whether the item should appear in bold.
ItemByIndex	Retrieves the handle of the item given its index in Items collection..
ItemCell	Retrieves the cell's handle given the item and the column.
ItemChild	Retrieves the child of a specified item.
ItemControlID	Retrieves the item's control identifier that was used by InsertControlItem.
ItemCount	Retrieves the number of items.
ItemData	Retrieves or sets the extra data for a specific item.
ItemDivider	Specifies whether the item acts like a divider item. The value indicates the index of column used to define the divider's title.
ItemDividerLine	Defines the type of line in the divider item.
ItemDividerLineAlignment	Specifies the alignment of the line in the divider item.
ItemFiltered	Checks whether the item is included in the control's filter.
ItemFont	Retrieves or sets the item's font.
ItemForeColor	Retrieves or sets a foreground color for a specific item.
ItemHasChildren	Adds an expand button to left side of the item even if the item has no child items.
ItemHeight	Retrieves or sets the item's height.
ItemItalic	Retrieves or sets a value that indicates whether the item should appear in italic.
ItemMaxHeight	Retrieves or sets a value that indicates the maximum height when the item's height is variable.
ItemMinHeight	Retrieves or sets a value that indicates the minimum height when the item's height is sizing.
ItemObject	Retrieves the ActiveX object associated, if the item was created using the InsertControlItem method.

ItemParent	Returns the handle of the item's parent item.
ItemPosition	Retrieves or sets a value that indicates the item's position in the children list.
ItemStrikeOut	Retrieves or sets a value that indicates whether the item should appear in strikeout.
ItemToIndex	Retrieves the index of item in the Items collection given its handle.
ItemUnderline	Retrieves or sets a value that indicates whether the item should appear in underline.
ItemWidth	Retrieves or sets a value that indicates the item's width while it contains an ActiveX control.
ItemWindowHost	Retrieves the window's handle that hosts an ActiveX control when the item was created using the InsertControlItem method.
ItemWindowHostCreateStyle	Retrieves or sets a value that indicates a combination of window styles used to create the ActiveX window host.
LastVisibleItem	Retrieves the handle of the last visible item.
LockedItem	Retrieves the handle of the locked item.
LockedItemCount	Specifies the number of items fixed on the top or bottom side of the control.
MatchItemCount	Retrieves the number of items that match the filter.
MergeCells	Merges a list of cells.
NextSiblingItem	Retrieves the next sibling of the item in the parent's child list.
NextVisibleItem	Retrieves the handle of next visible item.
PathSeparator	Returns or sets the delimiter character used for the path returned by the FullPath property.
PrevSiblingItem	Retrieves the previous sibling of the item in the parent's child list.
PrevVisibleItem	Retrieves the handle of previous visible item.
RemoveAllItems	Removes all items from the control.
RemoveItem	Removes a specific item.
RemoveSelection	Removes the selected items (including the descendents).
RootCount	Retrieves the number of root objects in the Items collection.

RootItem	Retrieves the handle of the root item giving its index in the root items collection.
SelectableItem	Specifies whether the user can select the item.
SelectAll	Selects all items.
SelectCount	Retrieves the handle of selected item giving its index in selected items collection.
SelectedItem	Retrieves the selected item's handle given its index in selected items collection.
Selection	Selects items by index.
SelectItem	Selects or unselects a specific item.
SelectPos	Selects items by position.
SetParent	Changes the parent of the given item.
SortableItem	Specifies whether the item is sortable.
SortChildren	Sorts the child items of the given parent item in the control. SortChildren will not recurse through the grid, only the immediate children of Item will be sorted.
SplitCell	Splits a cell, and returns the inner created cell.
UnmergeCells	Unmerges a list of cells.
UnselectAll	Unselects all items.
UnsplitCell	Unsplits a cell.
VisibleCount	Retrieves the number of visible items.
VisibleItemCount	Retrieves the number of visible items.

property Items.AcceptSetParent (Item as HITEM, NewParent as HITEM) as Boolean

Verifies whether the item can be the child of another item

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
NewParent as HITEM	A long expression that indicates the handle of the parent item.
Boolean	A boolean expression that indicates whether the Item can be child of the NewParent item.

The AcceptSetParent property doesn't change the parent item. Use the SetParent method to change the item's parent. Use the ItemParent property to retrieve the item's parent. Use the InsertItem method to add child items to another item. An item is called root, if it has no parent (ItemParent() gets 0).

method Items.AddItem ([Value as Variant])

Adds a new item, and returns a handle to the newly created item.

Type	Description
Value as Variant	A variant expression that indicates the cell's value for the first column or a safe array that holds the values for each column.

Return	Description
HITEM	A long expression that indicates the handle of the newly created item.

Use the AddItem property to add new items/cards that have no parent (usually when your control acts like a list or in CardView mode). Adding new items fails, if the control has no columns. Use the Add method to add new columns to the control. Use InsertItem to insert child items (usually when your control acts like a tree). When a new item is added to the Items collection, the control fires the ViewItemUpdate(exAddItem) event. If the control contains more than one column use the CellValue property to set the cell's value. Use the CellValueFormat property to specify whether the value contains HTML format or computed fields. If the control has no columns the AddItem method fails. Use Add method to insert new columns to the control. Use the LockedItemCount property to lock or unlock items to the top or bottom side of the control. Use the MergeCells method to combine one or more cells in a single cell. Use the SplitCell property to split a cell. Use the BeginUpdate and EndUpdate methods to maintain performance while adding new columns and items. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.CellBackColor([Item as Variant], [ColIndex as Variant]) as Color

Retrieves or sets the cell's background color.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key
Color	A color expression that indicates the cell's background color.

To change the background color for the entire item you can use ItemBackColor property. Use the ClearCellBackColor method to clear the cell's background color. Use the BackColor property to specify the control's background color. Use the CellForeColor property to specify the cell's foreground color. Use the ItemForeColor property to specify the item's foreground color. Use the SelectedItem property to specify whether an item is selected or unselected. Use the Def(exCellBackColor) property to specify the background color for all cells in the column. Use the Add method to add new skins to the control. You can define new skins and to use it to mark some cells, like in the following samples. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.CellBold([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that specifies whether the cell should appear in bold.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key
Boolean	A boolean expression that indicates whether the cell should appear in bold.

Use the CellBold property to bold a cell. Use the ItemBold property to specify whether the item should appear in bold. Use the HeaderBold property of the Column object to bold the column's caption. Use the CellItalic, CellUnderline or CellStrikeOut property to apply different font attributes to the cell. Use the ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.CellButtonAutoWidth([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell's button fits the cell's caption.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression indicating whether the cell's button fits the cell's caption

By default, the CellButtonAutoWidth property is False. The CellButtonAutoWidth property has effect only if the CellHasButton property is true. Use the Def property to specify that all buttons in the column fit to the cell's content. If the CellButtonAutoWidth property is False, the width of the button is the same as the width of the column. If the CellButtonAutoWidth property is True, the button area covers only the cell's caption. Use the CellValue property to specify the button's caption. Use the CellValueFormat property to assign an HTML caption to the button.

property Items.CellCaption ([Item as Variant], [ColIndex as Variant]) as String

Gets the cell's display value.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
String	A string expression that indicates the cell's value as it is displayed on the user interface.

The CellCaption property retrieves the cell's display value as it is displayed on the control's user interface. If the cell has no editor associated (no editor was assigned to the column and no editor was assigned to the cell), the CellCaption property gets the string representation of the cell's value. Use the CellValue property to change the cell's value. For instance, if a cell has a drop down list editor, the CellCaption property retrieves the caption of the predefined values. Use the CellImage property to assign an icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the CellPicture property to assign a custom size picture to a cell.

property Items.CellChecked (RadioGroup as Long) as HCELL

Retrieves the handle of the cell that is checked, given the radio group identifier.

Type	Description
RadioGroup as Long	A long expression that indicates the radio group identifier.
HCELL	A long expression that indicates the cell's handle. Use the CellItem property to retrieve the handle of the owner item.

A radio group contains a set of cells of radio types. Use the CellHasRadioButton property to set the cell of radio type. To change the state for a cell you can use the CellState property. To add or remove a cell to a given radio group you have to use CellHasRadioButton property. Use the CellRadioGroup property to add cells in the same radio group. The control fires the ViewItemStateStartChanging(exCheckItem) / ViewItemStateEndChanging(exCheckItem) event when the check box or radio button state is changed.

property Items.CellData([Item as Variant], [ColIndex as Variant]) as Variant

Specifies the cell's extra data.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A variant expression that indicates the cell's user data.

Use the CellData to associate an extra data to your cell. Use ItemData when you need to associate an extra data with an item. The CellData value is not used by the control, it is only for user use. Use the Data property to assign an extra data to a column. Use the SortType property to get sorted the column by the CellData property.

Note: The intersection of an item with a column defines a cell. Each cell is uniquely represented by its handle. The cell's handle is of HCELL type, that's equivalent with a long type. All properties of Items object that have two parameters *Item* and *ColIndex,* refer a cell.

property Items.CellEnabled([Item as Variant], [ColIndex as Variant]) as Boolean

Returns or sets a value that determines whether a cell can respond to user-generated events.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell is enabled or disabled.

Use the CellEnabled property to disable a cell. A disabled cell looks grayed. Use the EnableItem property to disable an item. Once that one cell is disabled it cannot be checked or clicked. Use the SelectableItem property to specify the user can select an item. To disable a column you can use Enabled property of the Column object.

Note: The intersection of an item with a column defines a cell. Each cell is uniquely represented by its handle. The cell's handle is of HCELL type, that's equivalent with a long type. All properties of Items object that have two parameters *Item* and *ColIndex,* refer a cell.

property Items.CellFont ([Item as Variant], [ColIndex as Variant]) as IFontDisp

Retrieves or sets the cell's font.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or the column's key.
IFontDisp	A Font object that indicates the item's font.

By default, the CellFont property is nothing. If the CellFont property is noting, the cell uses the item's font. Use the CellFont and ItemFont properties to specify different fonts for cells or items. Use the CellBold, CellItalic, CellUnderline, CellStrikeout, ItemBold, ItemUnderline, ItemStrikeout, ItemItalic or CellValueFormat to specify different font attributes. Use the Refresh method to refresh the control's content on the fly. Use the BeginUpdate and EndUpdate methods if you are doing multiple changes, so no need for an update each time a change is done. Use the ItemHeight property to specify the height of the item.

property Items.CellForeColor([Item as Variant], [ColIndex as Variant]) as Color

Retrieves or sets the cell's foreground color.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Color	A color expression that indicates the cell's foreground color

The CellForeColor property identifies the cell's foreground color. Use the ClearCellForeColor property to clear the cell's foreground color. Use the ItemForeColor property to specify the the item's foreground color. Use the Def(exCellForeColor) property to specify the foreground color for all cells in the column. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.CellFormatLevel([Item as Variant], [ColIndex as Variant]) as String

Specifies the arrangement of the fields inside the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
String	A CRD string expression that indicates the layout of the cell. The Index elements in the CRD string indicates the index of the column being displayed.

By default, the CellFormatLevel property is empty. If the CellFormatLevel property is empty, the cell displays it's caption. Use the CellValue property to assign a value to a cell. If the CellFormatLevel property is not empty, it indicates the layout being displayed in the cell's area. For instance, the CellFormatLevel = "1/2" indicates that the cell's area is vertically divided such as the up part displays the caption of the cell in the first column, and the down part displays the caption of the cell in the second column. The height of the item is NOT changed, after calling the CellFormatLevel property. Use the ItemHeight property to specify the height of the item. Use the DefaultItemHeight property to specify the default height of the items before inserting them. Use the Def(exCellFormatLevel) property to specify the layout for all cells in the same column. For instance, you can have a specify layout for some cells using the Def(exCellFormatLevel) property (by default it is applied to all cells in the column), and for other cells you can use the CellFormatLevel property to specify different layouts, or to remove the default layout. Use the FormatLevel property to arrange the columns in the control's header bar.

property Items.CellHAlignment ([Item as Variant], [ColIndex as Variant]) as AlignmentEnum

Retrieves or sets a value that indicates the alignment of the cell's caption.

Type	Description
Item as Variant	A long expression that indicates the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the cell's caption.

The CellHAlignment property aligns a particular cell. Use the Alignment property of the Column object to align all the cells in the column. Use the CellVAlignment property to align vertically the caption of the cell, when the item displays its content using multiple lines. Use the ClearCellHAlignment method to clear the cell's alignment previously set by the CellHAlignment property. If the CellHAlignment property is not set, the Alignment property of the Column object indicates the cell's alignment. If the cell belongs to the column that displays the hierarchy (TreeColumnIndex property), the cell can be aligned to the left or to the right. Use the Def(exCellDrawPartsOrder) property to specify the order of the drawing parts inside the cell.

property Items.CellHasButton([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has an associated push button.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell contains a button.

The caption of the push button is defined by the CellValue property. Use the Def property to assign buttons to all cells in the column. Use the CellButtonAutoWidth property to specify whether the buttons fit the cell's content. If you need multiple buttons inside the same cell, you can split the cell in multiple pieces and add a button to each piece. Use the SplitCell property to split a cell. Use the Background(exCellButtonUp) or Background(exCellButtonDown) property to change the visual appearance for the buttons in the control.

property Items.CellHasCheckBox([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has an associated checkbox.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell contains a check box button.

Use the CellState property to change the state of the cell of the check box type. The cell cannot display in the same time a radio and a check button. Use the CellHasRadioButton property to add a radio button to your cell. Use the PartialCheck property to enable partial check feature. Use the Def property to assign check boxes for all cells in the column. Use the CellImage property to add a single icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the CellPicture property to load a custom size picture to a cell. Use the FilterType property on exCheck to filter for checked or unchecked items. Use the Def(exCellDrawPartsOrder) property to specify the order of the drawing parts inside the cell.

property Items.CellHasRadioButton([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has an associated radio button.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell contains a radio button.

Use the CellState property to change the state of the cell of the radio type. The cell cannot display in the same time a radio and a check button. The control fires ViewItemStateStartChanging(exCheckItem) / ViewItemStateEndChanging(exCheckItem) event when the cell's state has been changed. Call the CellHasCheckBox property to add a check box to the cell. Use the CellRadioGroup property To group or ungroup cells of radio type. Use the Def property to assign radio buttons to all cells in the column. Use the CellImage property to add a single icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the CellPicture property to load a custom size picture to a cell. Use the Def(exCellDrawPartsOrder) property to specify the order of the drawing parts inside the cell

property Items.CellHyperLink ([Item as Variant], [ColIndex as Variant]) as Boolean

Specifies whether the cell is highlighted when the cursor mouse is over the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell is of hyper link type.

A cell that has CellHyperLink property to True, is a cell of hyper link type.

property Items.CellImage ([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value that indicates the index of icon to display in the cell..

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long value that indicates the index of the icon in Images collection. The Images collection is 1 based. 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.

The CellImage property assigns a single icon to a cell. Use the CellImage() = 0 to remove the cell's icon, that was previous assigned using the CellImage property . Use the CellImages property to assign multiple icons to a single cell. The icon's size is always 16 x 16. Use the CellPicture property to load a a picture of different size. Use the Images or ReplaceIcon method to load icons to the control. Use the ItemFromPoint property to retrieve the part of the control being clicked. Use the CellHasCheckBox property to add a check box to a cell. Use the CellHasRadioButton property to assign a radio button to a cell. Use the FilterType property on exImage to filter items by icons. Use the <img> HTML tag to insert icons inside the cell's caption. Use the Def(exCellDrawPartsOrder) property to specify the order of the drawing parts inside the cell.

property Items.CellImages ([Item as Variant], [ColIndex as Variant]) as Variant

Specifies an additional list of icons shown in the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A string expression that indicates the list of icons shown in the cell. For instance, the "1,2,3" indicates that the icons 1, 2, 3 are displayed in the cell.

The CellImage property assign a single icon to the cell. Instead if multiple icons need to be assigned to a single cell you have to use the CellImages property. The CellImages property takes a list of additional icons and display them in the cell. The list is separated by ',' and should contain numbers that represent indexes to Images list collection. Use the Images or ReplaceIcon method to assign icons at runtime. Use the Def(exCellDrawPartsOrder) property to specify the order of the drawing parts inside the cell.

property Items.CellItalic([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that specifies whether the cell should appear in italic.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell should appear in italic.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.CellItem (Cell as HCELL) as HITEM

Retrieves the handle of the item that is the owner of a specfic cell.

Type	Description
Cell as HCELL	A long expression that indicates the handle of a cell.
HITEM	A long expression that indicates the item's handle.

Use the CellItem property to retrieve the item's handle. Use the ItemCell property to gets the cell's handle given an item and a column. Most of the properties of the Items object that have parameters [Item as Variant], [ColIndex as Variant], could use the handle of the cell to identify the cell, instead the ColIndex parameter.

Note: The intersection of an item with a column defines a cell. Each cell is uniquely represented by its handle. The cell's handle is of HCELL type, that's equivalent with a long type. All properties of Items object that have two parameters *Item* and *ColIndex,* refer a cell.

property Items.CellMerge([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves or sets a value that indicates the index of the cell that's merged to.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A long expression that indicates the index of the cell that's merged with, a safe array that holds the indexes of the cells being merged.

Use the CellMerge property to combine two or more cells in the same item in a single cell. The data of the source cell is displayed in the new larger cell. All the other cells' data is not lost. Use the ItemDivider property to display a single cell in the entire item. Use the UnmergeCells method to unmerge the merged cells. Use the CellMerge property to unmerge a single cell. Use the MergeCells method to combine one or more cells in a single cell. Use the SplitCell property to split a cell. Use the Add method to add new columns to the control.

property Items.CellParent ([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves the parent of an inner cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item where the cell is, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Variant	A long expression that indicates the handle of the parent cell.

Use the CellParent property to get the parent of the inner cell. The SplitCell method splits a cell in two cells (the newly created cell is called inner cell). Use the InnerCell property to get the inner cell. Use the CellItem property to get the item that's the owner of the cell. Use the ItemCell property to get a master cell giving the handle of the item and the index of the column. The CellParent property gets 0 if the cell is the master cell, not an inner cell. The parent cell is always displayed to the left side of the cell. The inner cell (InnerCell) is displayed to the right side of the cell.

property Items.CellPicture ([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves or sets the cell's picture.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A Picture object that indicates the cell's picture. (A Picture object implements IPicture interface), 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.

The control can associate to a cell a check or radio button, an icon, multiple icons, a picture and a caption. Use the CellPicture property to associate a picture to a cell. You can use the CellPicture property when you want to display images with different widths into a cell. Use the CellImage property to associate an icon from Images collection. Use the CellImages property to assign multiple icons to a cell. Use the CellHasCheckBox property to add a check box to a cell. Use the CellHasRadioButton property to assign a radio button to a cell. The CellPictureWidth property specifies the width in pixels of the cell's picture. If it is not specified, the picture's size determines the width to paint the picture inside the cell. The CellPictureHeight property specifies the height in pixels of the cell's picture. If it is not specified, the picture's size determines the height to paint the picture inside the cell. Use the <img> built-in HTML tag to insert multiple custom size picture to the same cell. Use the Def(exCellDrawPartsOrder) property to specify the order of the drawing parts inside the cell.

property Items.CellPictureHeight ([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value that indicates the height of the cell's picture.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long expression that indicates the height of the cell's picture, or -1, if the property is ignored.

By default, the CellPictureHeight property is -1. Use the CellPicture property to assign a custom size picture to a cell. Use the CellImage or CellImages property to assign one or more icons to the cell. The CellPictureWidth property has effect on CellPicture property only. Use the CellPictureWidth property to specify the width of the cell's picture. The CellPictureWidth and CellPictureHeight properties specifies the size of the area where the cell's picture is stretched. If the CellPictureWidth and CellPictureHeight properties are -1 (by default), the cell displays the full size picture. If the CellPictureHeight property is greater than 0, it indicates the height of the area where the cell's picture is stretched. Use the ItemHeight property to specify the height of the item.

property Items.CellPictureWidth ([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value that indicates the width of the cell's picture.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long expression that indicates the width of the cell's picture, or -1, if the property is ignored.

By default, the CellPictureWidth property is -1. Use the CellPicture property to assign a custom size picture to a cell. Use the CellImage or CellImages property to assign one or more icons to the cell. The CellPictureWidth property has effect on CellPicture property only. Use the CellPictureHeight property to specify the height of the cell's picture. The CellPictureWidth and CellPictureHeight properties specifies the size of the area where the cell's picture is stretched. If the CellPictureWidth and CellPictureHeight properties are -1 (by default), the cell displays the full size picture. If the CellPictureWidth property is greater than 0, it indicates the width of the area where the cell's picture is stretched.

property Items.CellRadioGroup([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value indicating which radio group a cell is contained in.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long expression that identifies the cell's radio group.

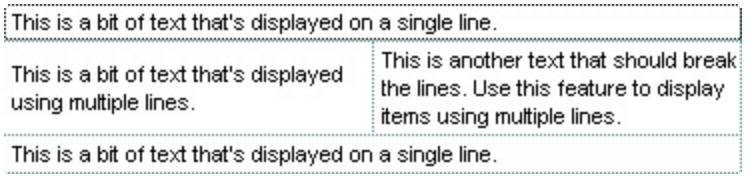
Use the CellRadioGroup property to add or remove a radio button from a group. In a radio group only one radio button can be checked. A radio cell cannot be contained by two different radio groups. Use the CellHasRadioButton property to add a radio button to a cell. When the cell's state is changed the control fires the ViewItemStateStartChanging(exCheckItem) / ViewItemStateEndChanging(exCheckItem) event. The CellState property specifies the cell's state. By default, when a cell of radio type is created the radio cell is not grouped to any of existent radio groups

property Items.CellSingleLine([Item as Variant], [ColIndex as Variant]) as CellSingleLineEnum

Retrieves or sets a value indicating whether the cell is painted using one line, or more than one line.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key
CellSingleLineEnum	A CellSingleLineEnum expression that indicates whether the cell displays its caption using one or more lines.

By default, the CellSingleLine property is exCaptionSingleLine / True, which indicates that the cell's caption is displayed on a single line. Use the Def(exCellSingleLine) property to specify that all cells in the column display their content using multiple lines. The control can displays the cell's caption using more lines, if the CellSingleLine property is exCaptionWordWrap or exCaptionBreakWrap. The CellSingleLine property wraps the cell's caption so it fits in the cell's client area. If the text doesn't fit the cell's client area, the height of the item is increased or decreased. When the CellSingleLine is exCaptionWordWrap / exCaptionBreakWrap / False, the height of the item is computed based on each cell caption. *If the CellSingleLine property is* exCaptionWordWrap / exCaptionBreakWrap / *False, changing the ItemHeight property has no effect.* Use the ItemMaxHeight property to specify the maximum height of the item when its height is variable. Use the CellVAlignment property to align vertically a cell.



If using the CellSingleLine / Def(exCellSingleLine) property, we recommend to set the ScrollBySingleLine property on True so all items can be scrolled.

property Items.CellSortData([Item as Variant], [ColIndex as Variant]) as Variant

Specifies the cell's sort data.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A long expression that indicates the cell's sort data.

The CellSortData property specifies the value being sorted if the SortType property is SortCellData or SortCellDataString. Use the CellData property to associate an extra data to a cell. Use the CellValue property to specify the cell's value. Use the CellCaption property to get the string being displayed in the cell.

Note: The intersection of an item with a column defines a cell. Each cell is uniquely represented by its handle. The cell's handle is of HCELL type, that's equivalent with a long type. All properties of Items object that have two parameters *Item* and *ColIndex,* refer a cell.

property Items.CellState([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets the cell's state. Affects only check and radio cells.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long value that indicates the cell's state.

Use the CellState property to change the cell's state. The CellState property has effect only for check and radio cells. Use the CellHasCheckBox property to assign a check box to a cell. Use the CellHasRadioButton property to add a radio button to a cell. The control fires the ViewItemStateStartChanging(exCheckItem) / ViewItemStateEndChanging(exCheckItem) event when user changes the cell's state. Use the PartialCheck property to allow partial check feature within the column. the FilterType property on exCheck to filter for checked or unchecked items.

property Items.CellStrikeOut([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that specifies whether the cell should appear in strikeout.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption.
Boolean	A boolean expression that indicates whether the cell should appear in strikeout.

If the CellStrikeOut property is True, the cell's font is displayed with a horizontal line through it. Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.CellToolTip([Item as Variant], [ColIndex as Variant]) as String

Retrieves or sets a value that indicates the cell's tool tip text.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
String	A string expression that indicates the cell's tooltip.

By default, the CellToolTip property is "..." (three dots). If the CellToolTip property is "..." the control displays the cell's caption if it doesn't fit the cell's client area. If the CellToolTip property is different than "...", the control shows a tooltip that displays the CellToolTip value. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ShowToolTip method to display a custom tooltip.

The tooltip 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 $_{subscript}$ 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:

- **<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:

Note: The intersection of an item with a column defines a cell. Each cell is uniquely represented by its handle. The cell's handle is of HCELL type, that's equivalent with a long type. All properties of Items object that have two parameters *Item* and *ColIndex,* that refers a cell.

property Items.CellUnderline([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that indicates whether the cell is underlined.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell is underlined.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula

property Items.CellVAlignment ([Item as Variant], [ColIndex as Variant]) as VAlignmentEnum

Retrieves or sets a value that indicates how the cell's caption is vertically aligned.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
VAlignmentEnum	A VAlignmentEnum expression that indicates the cell's vertically alignment.

The CellVAlignment property aligns vertically the cell. The CellVAligment property aligns the +/- sign if the item contains child items. The CellVAlignment property has effect if the item displays cells using multiple lines. Use the CellSingleLine property to wrap the cell's caption on multiple lines. Use the ItemHeight property to specify the height of the item. Use the **
** built-in HTML format to break a line, when CellValueFormat property is exHTML. Use the CellHAlignment property to align horizontally the cell. Use the Def(exCellVAlignment) property to specify the same vertical alignment for the entire column.

property Items.CellValue([Item as Variant], [ColIndex as Variant]) as Variant

Specifies the cell's value.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key. If the Item parameter is missing or it is zero (0), the ColIndex parameter is the handle of the cell being accessed.
Variant	A variant expression that indicates the cell's value or content. The cell's value supports built-in HTML format if the CellValueFormat property is exHTML.

Use the CellValue property to specify the value or the content for cells in the second, third columns and so on. The CellValueFormat property indicates the way the cell displays its content. The Def(exCellValueFormat) property indicates the format for all cells within the column.

The cell shows its text based on the CellValueFormat property as follows:

- **exText**, the CellValue indicates the text to be displayed without HTML formatting
- **exHTML**, the CellValue indicates the text to be displayed with HTML formatting, such as <b> to bold a portion of text.
- **exComputedField**, the CellValue property indicates a formula to display the cell's content based on the values of any cell in the current item. For instance, the %1 + %2 + %3 adds or concatenates the values from first 3 cells. The exComputedField can be combined with exHTML that indicates that the computed field may display HTML format. The ComputedField property specifies the formula to compute the entire column. The ComputeValue property can be used to get the result of specified formula.
- **exTotalField**, the CellValue indicates a formula to display the cell's content based on the values of any cell from any column, any item or its descendents. For instance, the sum(1,0,%1 + %2 + %3) gets the sum of first three columns from the direct descendents of the first item. The exTotalField can be combined with exHTML that indicates that the total field may display HTML format. The divider, unsortable or unselectable items do not count for total fields. The ComputeValue property can be used to get the result of specified formula.

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

Use the CellData property to associate an user data to a cell. The CellSortData property specifies the value being sorted if the SortType property is SortCellData or SortCellDataString. The AddItem or InsertItem method may specify the value for the first cell. Use the LockedItemCount property to lock or unlock items to the top or bottom side of the control. Use the ItemCell property to get the cell's handle based on the item and the column. Use the CellItem property to get the handle of the item that's the owner of the cell. Use the SplitCell property to split a cell.

property Items.CellValueFormat([Item as Variant], [ColIndex as Variant]) as ValueFormatEnum

Specifies how the cell's caption is displayed.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or column's key.
ValueFormatEnum	A long expression that defines the way how the cell's value is displayed. This value can be an OR combination of listed values. For instance, exHTML + exTotalField indicates a total field that may display HTML format

The component supports built-in HTML format. That means that you can use HTML tags when displays the cell's value . By default, the CellValueFormat property is exText. If the CellValueFormat is exText, the cell displays the CellValue property like it is. If the CellValueFormat is exHTML, the cell displays the CellValue property using the HTML tags specified in the ValueFormatEnum type. Use the Def property to specify whether all cells in the column display HTML format. Use the CellVAlignment property to align vertically a cell.

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

property Items.CellWidth([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value that indicates the width of the inner cell.

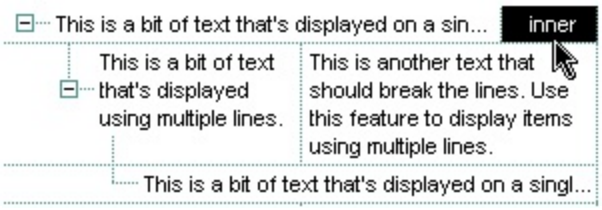
Type	Description
Item as Variant	A long expression that indicates the handle of the item where the cell is, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Long	A long expression that indicates the width of the cell.

The CellWidth property specifies the cell's width. The CellWidth property has effect only if the cell contains inner cells. The SplitCell method splits a cell in two cells (the newly created cell is called inner cell). Use the InnerCell property to get the inner cell. Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the item that's the owner of the cell. Use the BeginUpdate and EndUpdate methods to refresh the cell's width when changing it on the fly.

The CellWidth property specifies the width of the cell, where the cell is divided in two or multiple (inner) cells like follows:

- if the CellWidth property is less than zero, the master cell calculates the width of the inner cell, so all the inner cells with CellWidth less than zero have the same width in the master cell.
- if the CellWidth property is greater than zero, it indicates the width in pixels of the inner cell.

By default, the CellWidth property is -1, and so when the user splits a cell the inner cell takes the right half of the area occupied by the master cell.



property Items.ChildCount (Item as HITEM) as Long

Retrieves the number of children items.

Type	Description
Item as HITEM	A long expression that indicates the item's handle
Long	A long value that indicates the number of child items.

Use the ChildCount property to count the number of child items. Use the ItemChild property to get the handle of the first child item, if it exists. Use the ItemHasChildren property to built a virtual tree. A virtual tree loads items when the user expands an item. Use the ExpandItem property to expand or collapse an item. Use the InsertItem method to insert child items. Use the InsertControlItem method to insert child ActiveX controls.

method Items.ClearCellBackColor ([Item as Variant], [ColIndex as Variant])

Clears the cell's background color.

Type	Description
Item as Variant	An item's handle that indicates the owner of the cell.
ColIndex as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or column's key.

The ClearCellBackColor method clears the cell's background color when the CellBackColor property is used. Use the ItemBackColor property to specify the item's background color. Use the BackColor property to specify the control's background color

method Items.ClearCellForeColor ([Item as Variant], [ColIndex as Variant])

Clears the cell's foreground color.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or column's key.

The ClearCellForeColor method clears the cell's foreground color when CellForeColor property was used. Use the ItemForeColor property to specify the item's foreground color. Use the ForeColor property to specify the control's foreground color.

method Items.ClearCellHAlignment ([Item as Variant], [ColIndex as Variant])

Clears the cell's alignment.

Type	Description
Item as Variant	A long expression that indicates the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.

Use the ClearCellHAlignment method to clear the alignment of the cell's caption previously set using the CellHAlignment property. If the CellHAlignment property is not called, the Alignment property of the Column object specifies the alignment of the cell's caption.

method Items.ClearItemBackColor (Item as HITEM)

Clears the item's background color.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.

The ClearItemBackColor method clears the item's background color when ItemBackColor property was used. Use the BackColor property to specify the control's background color.

method Items.ClearItemForeColor (Item as HITEM)

Clears the item's foreground color.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.

The ClearItemForeColor method clears the item's foreground color when ItemForeColor property was used. Use the ForeColor property to specify the control's foreground color.

property Items.ComputeValue ([Expression as Variant], [Item as Variant], [ColIndex as Variant], [ValueFormatType as Variant]) as Variant

Computes the value of a specified formula.

Type	Description
Expression as Variant	A string expression that specifies the formula to compute
Item as Variant	A long expression that specifies the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.
ValueFormatType as Variant	A ValueFormatType expression that indicates the type of the formula being interpreted by the Expression parameter. For instance, if the ValueFormatType parameter is exTotalField, the Expression parameter should inidcate a total formula of type aggregate(list,direction,formula)
Variant	A string expression that indicates the result.

The ComputeValue property gets the result of a a computed or total field. The Item and ColIndex property refers the cells used as the source for the formula. Use the ComputeValue property to get the result of a total field. For instance, for a total field, the CellValue property indicates the formula, while the ComputeValue can be used to get the result of the formula at runtime.

The ComputeValue method returns the:

- value of the computed field, where the ValueFormatType is exComputedField, and the Expression indicates the formula for the computed field.
- value of the total field, where the ValueFormatType is exTotalField, and the Expression indicates a string as: aggregate(list,direction,formula)
- text with no HTML formatting, where the ValueFormatType is exHTML, and the Expression indicates the string including the HTML format.

For instance, based on the ValueFormatType and Expression parameters the result could be:

- exComputedField, dbl(%0) + dbl(%1), the sum between first two cells in the item referred by Item.
- exTotalField, sum(current,dir,dbl(%0) + dbl(%1)), the total of first two columns, for all direct child items of the item being referred by Item.
- exHTML, bold, returns bold (returns the result with no HTML formatting). In

this case, the Item and ColIndex have no effect.

property Items.DefaultItem as HITEM

Retrieves or sets a value that indicates the handle of the item used by Items properties in VFP.

Type	Description
HITEM	Retrieves the handle of the item that's used by all properties of Items object, that have a parameter Item.

The property is used in VFP implementation. The VFP fires "Invalid Subscript Range" error, while it tries to process a number greater than 65000. Since, the HITEM is a long value that most of the time exceeds 65000, the VFP users have to use this property, instead passing directly the handles to properties. The following sample shows to change the cell's image:

```
.Items.DefaultItem = .Items.AddItem("Item 1")
.Items.CellImage(0,1) = 2
```

In VFP the following sample fires: "Invalid Subscript Range":

```
i = .Items.AddItem("Item 1")
.Items.CellImage(i,1) = 2
```

because the i variable is grater than 65000.

So, if you pass zero to a property that has a parameter titled Item, the control takes instead the DefaultItem value.

property Items.EnableItem(Item as HITEM) as Boolean

Returns or sets a value that determines whether a item can respond to user-generated events.

Type	Description
Item as HITEM	A long expression that indicates the item's handle
Boolean	A boolean expression that indicates whether the item is enabled or disabled.

Use the EnableItem property to disable an item. A disabled item looks grayed and it is selectable. Use the SelectableItem property to specify the user can select an item. Once that an item is disabled all the cells of the item are disabled, so CellEnabled property has no effect. To disable a column you can use Enabled property of a Column object.

method Items.EnsureVisibleItem (Item as HITEM)

Ensures that the given item is in the visible client area.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.

The EnsureVisibleItem scrolls the control's content until the item fits the visible client area. The EnsureVisibleItem method expands the parent items. Use the IsItemVisible property to check if an item fits the control's client area. Use the EnsureVisibleColumn method to scroll the control's content so a column fits the control's client area. Use the Scroll method to scroll the control's client area by code. The EnsureVisibleItem method should not be called during BeginUpdate and EndUpdate methods. The EnsureOnSort property prevents scrolling the control's content when the user sorts items.

property Items.ExpandItem(Item as HITEM) as Boolean

Expands, or collapses, the child items of the specified item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being expanded or collapsed.
Boolean	A boolean expression that indicates whether the item is expanded or collapsed.

Use ExpandItem property to programmatically expand or collapse an item. Use the ExpandItem property to check whether an items is expanded or collapsed. To check if the item has child items you can use ChildCount property. Use the ItemHasChildren property to display a +/- expand sign to the item even if it doesn't contain child items. The ExpandOnSearch property specifies whether the control expands nodes when incremental searching is on (AutoSearch property is different than 0) and user types characters when the control has the focus. Use the ExpandOnKeys property to specify whether the user expands or collapses the focused items using arrow keys.

property Items.FindItem (Value as Variant, [ColIndex as Variant], [StartIndex as Variant]) as HITEM

Finds an item, looking for Caption in ColIndex colum. The searching starts at StartIndex item.

Type	Description
Value as Variant	A Variant expression that indicates the caption that is searched for.
ColIndex as Variant	A string expression that indicates the column's caption, or a long expression that indicates the column's index.
StartIndex as Variant	A long value that indicates the index of item from where the searching starts.
HITEM	A long expression that indicates the item's handle that matches the criteria.

Use the FindItem to search for an item. Finds a control's item that matches CellValue(Item, ColIndex) = Caption. The StartIndex parameter indicates the index from where the searching starts. If it is missing, the searching starts from the item with the 0 index. The searching is case sensitive only if the ASCIIUpper property is empty. Use the AutoSearch property to enable incremental search feature within the column.

property Items.FindItemData (UserData as Variant, [StartIndex as Variant]) as HITEM

Finds the item giving its data.

Type	Description
UserData as Variant	A variant value that indicates the value being searched
StartIndex as Variant	A long expression that indicates the handle of the item where the searching starts
HITEM	A long expression that indicates the handle of the item found.

Use the FindItemData property to search for an item giving its extra-data. Use the ItemData property to associate an extra data to an item. Use the FindItem property to locate an item given its caption. Use the FindPath property to search for an item given its path.

property Items.FindPath (Path as String) as HITEM

Finds an item given its path.

Type	Description
Path as String	A string expression that indicates the item's path
HITEM	A long expression that indicates the item's handle that matches the criteria.

The FindPath property searches the item on the column SearchColumnIndex. Use the FullPath property in order to get the item's path. Use the FindItem to search for an item.

property Items.FirstVisibleItem as HITEM

Retrieves the handle of the first visible item in control.

Type	Description
HITEM	A long expression that indicates the item's handle that indicates the first visible item.

Use the FirstVisibleItem, NextVisibleItem and IsItemVisible properties to get the items that fit the client area. Use the NextVisibleItem property to get the next visible item. Use the IsVisibleItem property to check whether an item fits the control's client area. Use the RootItem property to get the first visible item in the list. The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use the PrevVisibleItem property to retrieve the previous visible item.

property Items.FocusItem as HITEM

Retrieves the handle of item that has the focus.

Type	Description
HITEM	A long expression that indicates the item's handle that is focused.

If there is no focused item the FocusItem property retrieves 0. At one moment, only one item can be focused. When the selection is changed the focused item is changed too. Use the SelectItem property to select a new item. the FocusItem property gets the selected item too. Use the SingleSel property to specify whether the control supports single or multiple selection. Use the ShowFocusRect property to indicate whether the control draws a marking rectangle around the focused item. You can change the focused item, by selecting a new item using the SelectItem method. If the items is not selectable, it is not focusable as well. Use the SelectableItem property to specify whether an item is selectable/focusable.

property Items.FormatCell([Item as Variant], [ColIndex as Variant]) as String

Specifies the custom format to display the cell's content.

Type	Description
Item as Variant	A long expression that indicates the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.
String	A string expression that indicates the format to be applied on the cell's value, including HTML formatting, if the cell supports it.

By default, the FormatCell property is empty. The format is being applied if valid (not empty, and syntactically correct). The expression may be a combination of variables, constants, strings, dates and operators, and value. The *value* operator gives the value to be formatted. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by # character, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM. The FormatColumn property applies the predefined format for all cells in the columns. The CellValue property indicates the cell's value.

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

For instance:

- the "*currency(value)*" displays the column using the current format for the currency ie, 1000 gets displayed as $1,000.00
- the "*longdate(date(value))*" converts the value to a date and gets the long format to display the date in the column, ie #1/1/2001# displays instead Monday, January 01, 2001
- the "*'' + ((0:=proper(value)) left 1) + '' + (=:0 mid 2)*" converts the name to proper, so the first letter is capitalized, bolds the first character, and let unchanged the rest, ie a "mihai filimon" gets displayed "**M**ihai Filimon".
- the "*len(value) ? ((0:=dbl(value)) < 10 ? '<fgcolor=808080><font ;7>' : '') + currency(=:0)*" displays the cells that contains not empty daya, the value in currency format, with a different font and color for values less than 10, and bolded for those that are greater than 10, as can see in the following screen shot in the column (A+B+C):

The **value** keyword in the FormatColumn property indicates the value to be formatted.

The expression supports cell's identifiers as follows:

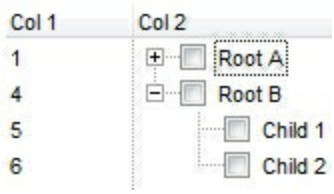
- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

The predefined operators for auto-numbering are:

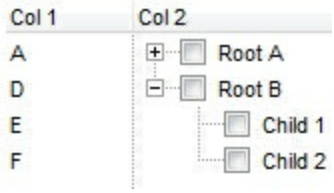
- number **index** 'format', indicates the index of the item. The first added item has the index 0, the second added item has the index 1, and so on. The index of the item remains the same even if the order of the items is changed by sorting. For instance, 1 index '' gets the index of the item starting from 1 while 100 index '' gets the index of the item starting from 100. The number indicates the starting index, while the format is a set of characters to be used for specifying the index. If the format is missing, the index of the item is formatted as numbers. For instance: 1 index 'A-Z' gets the index as A, B, C... Z, BA, BB, ... BZ, CA, ... The 1 index 'abc' gives the index as: a,b,c,ba,bb,bc,ca,cb,cc,.... You can use other number formatting function to format the

returned value. For instance "1 index " format '0||2|:'" gets the numbers grouped by 2 digits and separated by : character.

In the following screen shot the FormatColumn("Col 1") = "1 index ""

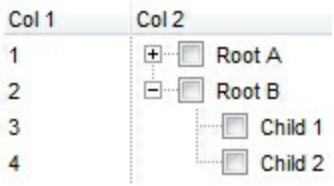


In the following screen shot the FormatColumn("Col 1") = "1 index 'A-Z'"

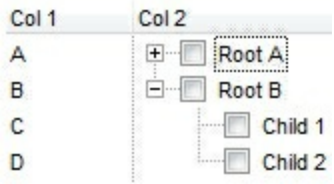


- number **apos** 'format' indicates the absolute position of the item. The first displayed item has the absolute position 0 (scrolling position on top), the next visible item is 1, and so on. The number indicates the starting position, while the format is a set of characters to be used for specifying the position. For instance, 1 apos " gets the absolute position of the item starting from 1, while 100 apos " gets the position of the item starting from 100. If the format is missing, the absolute position of the item is formatted as numbers.

  In the following screen shot the FormatColumn("Col 1") = "1 apos ""



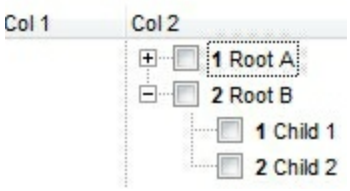
  In the following screen shot the FormatColumn("Col 1") = "1 apos 'A-Z'"



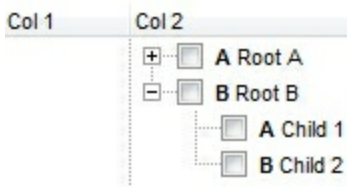
- number **pos** 'format' indicates the relative position of the item. The relative position is the position of the visible child item in the parent children collection. The number indicates the starting position, while the format is a set of characters to be used for specifying the position. For instance, 1 pos " gets the relative position of the item

starting from 1, while 100 pos '' gets the relative position of the item starting from 100. If the format is missing, the relative position of the item is formatted as numbers. *The difference between pos and opos can be seen while filtering the items in the control. For instance, if no filter is applied to the control, the pos and opos gets the same result. Instead, if the filter is applied, the opos gets the position of the item in the list of unfiltered items, while the pos gets the position of the item in the filtered list.*

In the following screen shot the FormatColumn("Col 2") = "'' + 1 pos '' + ' ' + value"

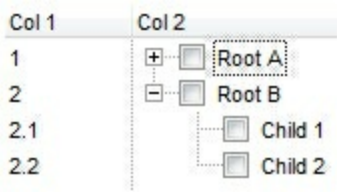


In the following screen shot the FormatColumn("Col 2") = "'' + 1 pos 'A-Z' + ' ' + value"

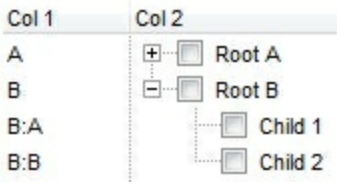


- number **opos** 'format' indicates the relative old position of the item. The relative old position is the position of the child item in the parent children collection. The number indicates the starting position, while the format is a set of characters to be used for specifying the position.For instance, 1 pos '' gets the relative position of the item starting from 1, while 100 pos '' gets the relative position of the item starting from 100. If the format is missing, the relative position of the item is formatted as numbers. *The difference between pos and opos can be seen while filtering the items in the control. For instance, if no filter is applied to the control, the pos and opos gets the same result. Instead, if the filter is applied, the opos gets the position of the item in the list of unfiltered items, while the pos gets the position of the item in the filtered list.*
- number **rpos** 'format' indicates the relative recursive position of the item. The recursive position indicates the position of the parent items too. The relative position is the position of the visible child item in the parent children collection. The number indicates the starting position, while the format is of the following type "delimiter|format|format|...". If the format is missing, the delimiter is . character, and the positions are formatted as numbers. The format is applied consecutively to each parent item, from root to item itself.

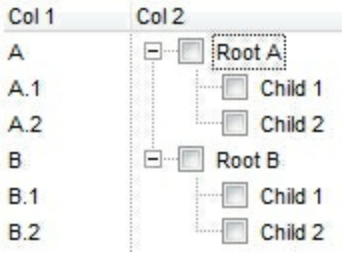
  In the following screen shot the FormatColumn("Col 1") = "1 rpos ""



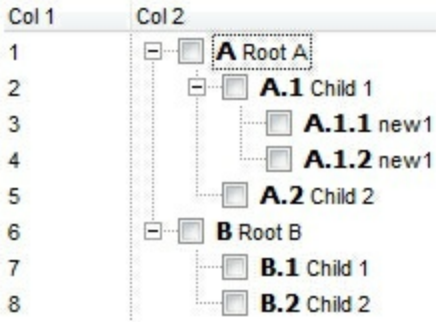
In the following screen shot the FormatColumn("Col 1") = "1 rpos ':|A-Z'"



In the following screen shot the FormatColumn("Col 1") = "1 rpos '.|A-Z|'"



In the following screen shot the FormatColumn("Col 1") = "1 apos "" and FormatColumn("Col 2") = "'' + 1 rpos '.|A-Z|' + ' ' + value"



- number **rindex** 'format', number **rapos** 'format' and number **ropos** 'format' are working similar with number **rpos** 'format', excepts that they gives the index, absolute position, or the old child position.

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

property Items.FullPath (Item as HITEM) as String

Returns the fully qualified path of the referenced item in an ExView control.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
String	A string expression that indicates the fully qualified path.

Use the FullPath property in order to get the fully qualified path of the referenced item. Use PathSeparator to change the separator used by FullPath property. Use the FindPath property to get the item's selected based on its path. The fully qualified path is the concatenation of the text in the given cell's caption property on the column SearchColumnIndex with the CellValue property values of all its ancestors.

property Items.GroupItem (Item as HITEM) as Long

Indicates a group item if positive, and the value specifies the index of the column that has been grouped.

Type	Description
Item as HITEM	A Long expression that specifies the handle of the item being queried
Long	A Long expression that specifies index of the column being grouped, or a negative value if the item is a regular item, not a grouping item.

The GroupItem method determines the index of the column that indicates the column being grouped. In other words, the CellCaption(Item,GroupItem(Item)) gets the default caption to be displayed for the grouping item. The Ungroup method removes all grouping items. For instance, when a column gets grouped by, the control sorts by that column, collects the unique values being found, and add a new item for each value found, by adding the items of the same value as children. The (ViewItemUpdate(exAddGroupItem) event is fired for each new item to be inserted in the Items collection during the grouping.

property Items.InnerCell ([Item as Variant], [ColIndex as Variant], [Index as Variant]) as Variant

Retrieves the inner cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item where the cell is, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Index as Variant	A long expression that indicates the index of the inner being requested. If the Index parameter is missing or it is zero, the InnerCell property retrieves the master cell.
Variant	A long expression that indicates the handle of the inner cell.

Use the InnerCell property to get the inner cell. The InnerCell(, , 0) property always retrieves the same cell. The InnerCell(, , 1) retrieves the first inner cell, and so on. The InnerCells property always retrieves a non empty value. For instance, if a cell contains only two splitted cells, the InnerCell(, , 3), or InnerCell(, , 4), and so on, always retrievs the last inner cell. The SplitCell method splits a cell in two cells (the newly created cell is called inner cell). Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the item that's the owner of the cell. Use the CellWidth property to specify the width of the inner cell. Use the CellParent property to determine whether the cell is a master cell or an inner cell. If the CellParent property gets 0, it means that the cell is master, else it is inner.

method Items.InsertControlItem (Parent as HITEM, ControlID as String, [License as Variant])

Inserts a new item of ActiveX type, and returns a handle to the newly created item.

Type	Description
Parent as HITEM	A long expression that indicates the handle of the parent item where the ActiveX will be inserted. If the argument is missing then the InsertControlItem property inserts the ActiveX control as a root item. If the Parent property is referring a locked item (ItemLocked property), the InsertControlItem property doesn't insert a new child ActiveX, instead insert the ActiveX control to the locked item that's specified by the Parent property.
ControlID as String	A string expression that can be formatted as follows: a prog ID, a CLSID, a URL, a reference to an Active document , a fragment of HTML.
License as Variant	A string expression that indicates the runtime license key for the component being inserted, if required. Only, the vendor of the component you are going to use is able to give you such of runtime license, so please contact the control's vendor for such of key. Your development license key is not compatible with the runtime license key, so it can't be used here.

Return	Description
HITEM	A long expression that indicates the item's handle that indicates the newly created item.

The control supports ActiveX hosting, so you can insert any ActiveX component as a child item of the control. If you are using the /NET assembly you can use the InsertObjectItem property to insert a /NET control as a child item of the control. The InsertControlItem property creates the specified ActiveX control and hosts to a new child item of the control, while the InsertObjectItem property hosts the already created object to a new child item of the control.

method Items.InsertItem ([Parent as HITEM], [UserData as Variant], [Value as Variant])

Inserts a new item, and returns a handle to the newly created item.

Type	Description
Parent as HITEM	A long expression that indicates the item's handle that indicates the parent item where the newly item is inserted
UserData as Variant	A Variant expression that indicates the item's extra data. Use the ItemData property to retrieve later this value.
Value as Variant	A Variant expression that indicates the cell's value on the first column, or a safe array that holds values for each column.

Return	Description
HITEM	Retrieves the handle of the newly created item.

Use the InsertItem property to add a new child to an item. The InsertItem property fires the ViewItemUpdate(exAddItem) event. You can use the InsertItem(,,"Root") or AddItem("Root") to add a root item. An item that has no parent is a root item. To insert an ActiveX control, use the InsertControlItem property of the Items property. Use the CellValue property to specify the values for cells in the second, third columns, and so on. Use the CellValueFormat property to specify whether the value contains HTML format or computed fields. Use the LockedItemCount property to lock or unlock items to the top or bottom side of the control. Use the MergeCells method to combine one or more cells in a single cell. Use the SplitCell property to split a cell. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

method Items.InsertObjectItem (Parent as HITEM, [UserData as Variant], [Obj as Variant])

Inserts a new item that hosts the giving object, and returns a handle to the newly created item.

Type	Description
Parent as HITEM	A long expression that indicates the handle of the parent item where the object will be inserted. If the argument is missing then the InsertObjectItem property inserts the object as a root item. If the Parent property is referring a locked item (ItemLocked property), the InsertObjectItem property doesn't insert a new child, instead places the object to the locked item that's specified by the Parent property.
UserData as Variant	A VARIANT expression being specified at creating time, which can be accessed during the ViewItemUpdate(exAddItem) event. The ItemData property indicates the extra data associated with any item. The ItemData property is initalized with the value of the UserData parameter.
Obj as Variant	A object being hosted. The most common type is System.Windows.Forms.Control from the /NET framework. Generally, the Obj could be any control that can be placed to a form or dialog and it is visible at runtime. The Obj can not be a windowless control (a control that does not require a window, such a line or circle). The Obj parameter could be also an ActiveX control (that has already being placed in the form/dialog) in this case, the Obj should be the result of the property Object() (VB6, VFP). GetOcx() property. Finally, the Obj parameter could be of long type (numeric) in which case it should refer the handle of a window that follows to be hosted in the newly created item. The handle of the window can be obtained as m_hWnd member of MFC classes, hWnd or Handle property in the /NET framework. After creating the host, the ItemObject property can be used to retrieve the originally object (Obj parameter).

Return	Description
HITEM	A long expression that indicates the item's handle that indicates the newly created item.

The control supports /NET Control hosting, so you can insert any /NET component as a child item of the control. This property is provided for the /NET assembly, but it is available for the /COM environment too. The InsertObjectItem property hosts the already created object to a new child item of the control while the InsertControlItem property creates the specified ActiveX control and hosts to a new child item of the control. So, the difference between the InsertObjectItem and InsertControlItem is that the InsertObjectItem does not create the object, while the InsertControlItem creates the specified control. If you are using the /NET assembly, the Obj should be the object to be inserted (usually of System.Windows.Forms.Control type), while for the /COM environment, the Obj should be the ActiveX control being already placed to a form, or a long expression that specifies the handle of the window to be hosted in a new child item of the control.

- The ItemHeight property specifes the height of the item, and so the height of the hosted object.
- The ItemWidth property specifies the width of hosted object, or the position/column in the item where the object is displayed.
- The ItemAllowSizing property indicates whether the user can resize the item at runtime, and so the object being hosted.
- The ItemObject property retrieves the originally object if the item was previously created using the InsertObjectItem property, or the created ActiveX control if using the InsertControlItem property.

property Items.IsItemLocked (Item as HITEM) as Boolean

Returns a value that indicates whether the item is locked or unlocked.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Boolean	A boolean expression that indicates whether the item is locked or unlocked.

Use the IsItemLocked property to check whether an item is locked or unlocked. A locked item is always displayed on the top or bottom side of the control no matter if the control's list is scrolled up or down. Use the LockedItemCount property to add or remove items fixed/locked to the top or bottom side of the control. Use the LockedItem property to access a locked item by its position. Use the ShowLockedItems property to show or hide the locked items.

property Items.IsItemVisible (Item as HITEM, [Partially as Variant]) as Boolean

Checks if the specific item fits the control's client area.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Partially as Variant	A boolean expression that indicates whether the item is partially visible or not. If the Partially parameter is missing, the True value is used.
Boolean	A boolean expression that indicates whether the item fits the client area.

To make sure that an item fits the client area call EnsureVisibleItem method. Use the FirstVisibleItem, NextVisibleItem and IsItemVisible properties to get the items that fit the client area. Use the NextVisibleItem property to get the next visible item. Use the IsVisibleItem property to check whether an item fits the control's client area.

property Items.ItemAllowSizing(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether a user can resize the item at run-time.

Type	Description
Item as HITEM	A HITEM expression that indicates the handle of the item that can be resized.
Boolean	A Boolean expression that specifies whether the user can resize the item at run-time.

By default, the user can resize the item at run-time using mouse movements. Use the ItemAllowSizing property to specify whether a user can resize the item at run-time. Use the ItemsAllowSizing property to specify whether all items are resizable or not. Use the ItemHeight property to specify the height of the item. An item is resizable if the ItemAllowSizing property is True, or if the ItemsAllowSizing property is True (that means all items are resizable), and the ItemAllowSizing property is not False. For instance, if your application requires all items being resizable but only few of them being not resizable, you can have the ItemsAllowSizing property on True, and for those items that are not resizable, you can call the ItemAllowSizing property on False. The user can resize an item by moving the mouse between two items, so the vertical split cursor shows up, click and drag the mouse to the new position. Use the CellSingleLine property to specify whether the cell displays its caption using multiple lines. The ScrollBySingleLine property is automatically set on True, as soon as the user resizes an item.

property Items.ItemAppearance(Item as HITEM) as AppearanceEnum

Specifies the item's appearance while it's of ActiveX type.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that was previously created by InsertControlItem property.
AppearanceEnum	An AppearanceEnum value that indicates the item's appearance.

Use the ItemAppearance property to specify the item's appearance if the item is of ActiveX type. Use the InsertControlItem property to insert an ActiveX control inside. Use the ItemObject property to access the object being created by the InsertControlItem property. Use the ItemHeight property to specify the height of the item when containing an ActiveX control.

property Items.ItemBackColor(Item as HITEM) as Color

Retrieves or sets a background color for a specific item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Color	A color expression that indicates the item's background color.

Use the CellBackColor property to change the cell's background color. To change the background color of the entire control you can call BackColor property of the control. Use the ClearItemBackColor property to clear the item's background color, after setting using the ItemBackColor property. Use the SelBackColor property to change appearance for the selected items. The HTML colors are not applied if the item is selected. Use the SelectedItem property to specify whether an item is selected or unselected. Use the Add method to add new skins to the control. You can define new skins and to use it to mark some items, like in the following samples. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.ItemBold(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item is bolded.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the item is bolded.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.ItemByIndex (Index as Long) as HITEM

Retrieves the handle of the item given its index in the Items collection..

Type	Description
Index as Long	A long value that indicates the item's index.
HITEM	A long expression that indicates the item's handle

Use the ItemByIndex to get the index of an item. Use the ItemCount property to count the items in the control. the Use the ItemPosition property to get the item's position. Use the ItemToIndex property to get the index of giving item. For instance, The ItemByIndex property is the default property for Items object, so the following statements are equivalents: View1.Items(0), View1.Items.ItemByIndex(0).

property Items.ItemCell (Item as HITEM, ColIndex as Variant) as HCELL

Retrieves the cell's handle given the item and the column.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the the column's index, a string expression that indicates the column's caption or the column's key.
HCELL	A long value that indicates the cell's handle.

The ItemCell property retrieves the handle of the cell that belongs to the item on the specified column. The InnerCell properties always returns the handle to the master cells (master cell is a cell where the splitting starts). Use the SplitCell property to split a cell into multiple cells. Use the MergeCells property to merge multiple cells.

property Items.ItemChild (Item as HITEM) as HITEM

Retrieves the first child item of a specified item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the item's handle that indicates the first child item of the Item

If the ItemChild property gets 0, the item has no child items. Use the ItemChild property to get the first child of an item. The NextVisibleItem or NextSiblingItem gets the next visible, sibling item. Use the ChildCount property to count the number of child items. Use the ItemHasChildren property to built a virtual grid. A virtual grid loads items when the user expands an item. Use the ItemParent property to retrieve the handle of the parent item. The control displays a +/- sign to parent items, if the HasButtons property is not zero, the ItemChild property is not empty, or the ItemHasChildren property is True.

property Items.ItemControlID (Item as HITEM) as String

Retrieves the item's control identifier that was used by InsertControlItem property.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
String	A string expression that indicates the control identifier used by InsertControlItem property to create an item that hosts an ActiveX control.

The ItemControlID property retrieves the control identifier used by the InsertControlItem property. If the item was created using AddItem or InsertItem properties the ItemControlID property retrieves an empty string. For instance, the ItemControlID property can be used to check if an item contains an ActiveX control or not.

property Items.ItemCount as Long

Retrieves the number of items.

Type	Description
Long	A long value that indicates the number of items into Items collection

The ItemCount property counts the items in the control. Use the ItemByIndex property to access an item giving its index. Use the AddItem, InsertItem, InsertControlItem, DataSource property to add new items to the control. Use ChildCount to get the number of child items.

property Items.ItemData(Item as HITEM) as Variant

Retrieves or sets the extra data for a specific item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Variant	A variant value that indicates the item's extra data.

Use the ItemData property to assign an extra value to an item. Use CellData property to associate an extra data with a cell. The ItemData and CellData are of Variant type, so you will be able to save here what ever you want: numbers, objects, strings, and so on. The user data is only for user use. The control doesn't use this value. Use the Data property to assign an extra data to a column.

property Items.ItemDivider(Item as HITEM) as Long

Specifies whether the item acts like a divider or normal item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long expression that indicates the column's index.

A divider item uses the item's client area to display a single cell. You can use the ItemDivider property to separate the items, display groups of items or display total or subtotals fields. The ItemDivider property specifies the index of the cell being displayed in the item's client area. In other words, the divider item merges the item cells into a single cell. The CellHAlignment property specifies the horizontal alignment for the cell's content. Use the ItemDividerLine property to define the line that underlines the divider item. Use the LockedItemCount property to lock items on the top or bottom side of the control. Use the MergeCells method to combine two or multiple cells in a single cell. Use the SelectableItem property to specify the user can select an item. A divider item has sense for a control with multiple columns. The SortableItem property specifies whether the item keeps its position after sorting.

property Items.ItemDividerLine(Item as HITEM) as DividerLineEnum

Defines the type of line in the divider item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
DividerLineEnum	A DividerLineEnum expression that indicates the type of the line in the divider item.

By default, the ItemDividerLine property is SingleLine. The ItemDividerLine property specifies the type of line that underlines a divider item. Use the ItemDivider property to define a divider item. Use the ItemDividerLine and ItemDividerAlignment properties to define the style of the line into the divider item. Use the CellMerge property to merge two or more cells.

property Items.ItemDividerLineAlignment(Item as HITEM) as DividerAlignmentEnum

Specifies the alignment of the line in the divider item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle
DividerAlignmentEnum	A DividerAlignmentEnum expression that specifies the line's alignment.

By default, the ItemDividerLineAlignment property is DividerBottom. The Use the ItemDividerLine and ItemDividerLineAlignment properties to define the style of the line into a divider item. Use the ItemDivider property to define a divider item.

property Items.ItemFiltered (Item as HITEM) as Boolean

Checks whether the item is included in the control's filter.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item
Boolean	A boolean expression that indicates whether the item is filtered.

Use the ItemFiltered property to check whether an item is included in the control's filter. Use the FilterType property to specify the type of filter that's applied to a column. The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property. The ItemCount property counts the items in the control's list. Use the ItemByIndex property to access an item giving its index.

property Items.ItemFont (Item as HITEM) as IFontDisp

Retrieves or sets the item's font.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
IFontDisp	A font object being used.

By default, the ItemFont property is nothing. If the ItemFont property is nothing, the item uses the control's font. Use the ItemFont property to define a different font for the item. Use the CellFont and ItemFont properties to specify different fonts for cells or items. Use the CellBold, CellItalic, CellUnderline, CellStrikeout, ItemBold, ItemUnderline, ItemStrikeout, ItemItalic or CellValueFormat to specify different font attributes. Use the ItemHeight property to specify the height of the item. Use the Refresh method to refresh the control's content on the fly. Use the BeginUpdate and EndUpdate methods if you are doing multiple changes, so no need for an update each time a change is done. Use the ItemHeight property to specify the height of the item.

property Items.ItemForeColor(Item as HITEM) as Color

Retrieves or sets a foreground color for a specific item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Color	A color expression that defines the item's foreground color

Use the CellForeColor property to change the item's foreground color. Use the ForeColor property to change the control's foreground color. Use the ClearItemForeColor property to clear the item's foreground color. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.ItemHasChildren (Item as HITEM) as Boolean

Adds an expand button to left side of the item even if the item has no child items.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the control adds an expand button to the left side of the item even if the item has no child items.

By default, the ItemHasChidren property is False. Use the ItemHasChildren property to build a virtual grid. Use the ViewItemStateStartChanging(exExpandItem) event to add new child items to the expanded item. Use the ItemChild property to get the first child item, if exists. Use the ItemChild or ChildCount property to determine whether an item contains child items. The control displays a +/- sign to parent items, if the HasButtons property is not empty, the ItemChild property is not empty, or the ItemHasChildren property is True. Use the InsertItem method to insert a new child item. Use the CellData or ItemData property to assign an extra value to a cell or to an item.

property Items.ItemHeight(Item as HITEM) as Long

Retrieves or sets the item's height.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Long	A long value that indicates the item's height.

To change the default height of the item before inserting it into the items collection you can call DefaultItemHeight property. The control supports items with different heights. When an item hosts an ActiveX control (was previously created by the InsertControlItem property), the ItemHeight property changes the height of contained ActiveX control too. *If the CellSingleLine property is False, the ItemHeight property has **no** effect. The Column.Def(exCellPaddingTop) and Column.Def(exCellPaddingBottom) defines the vertical padding.* Use the ItemMaxHeight property to specify the maximum height of the item when its height is variable. Use the CellVAlignment property to align vertically a cell. Use the ScrollBySingleLine property when using items with different heights. Use the SelectableItem property to specify whether the user can select an item. For instance, in order to hide an item you can set the ItemHeight property on 0, and SelectableItem property on False. Use the ItemAllowSizing property to specify whether the user can resize the item at runtime.

property Items.ItemItalic(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in italic.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the item's font attributes include Italic attribute.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.ItemMaxHeight(Item as HITEM) as Long

Retrieves or sets a value that indicates the maximum height when the item's height is variable.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item. If the Item is 0, setting the ItemMaxHeight property changes the maximum-height for all items. For instance, the ItemMaxHeight(0) = 24, changes the maximum height for all items to be 24 pixels wide.
Long	A long value that indicates the maximum height when the item's height is variable.

By default, the ItemMaxHeight property is -1. The ItemMaxHeight property has effect only if it is greater than 0, and item contains cells with CellSingleLine property on False. Use the ItemHeight property to get the item's height. Use the CellVAlignment property to align vertically a cell. Use the DefaultItemHeight property to specify the default height for all items before loading items.

property Items.ItemMinHeight(Item as HITEM) as Long

Retrieves or sets a value that indicates the minimum height when the item's height is sizing.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item. If the Item is 0, setting the ItemMinHeight property changes the minimum-height for all items. For instance, the ItemMinHeight(0) = 24, changes the minimum height for all items to be 24 pixels wide.
Long	A long value that indicates the minimum height when the item's height is variable.

By default, the ItemMinHeight property is -1. The ItemMinHeight property has effect only if the item contains cells with CellSingleLine property on False. The ItemMaxHeight property specifies the maximum height of the item while resizing. The CellSingleLine property specifies whether a cell displays its caption using multiple lines. The ItemHeight property has no effect, if the CellSingleLine property is False. If the CellSingleLine property is False, you can specify the minimum height for the item using the ItemMinHeight property. Use the ItemAllowSizing property to specify whether the user can resize the item at runtime.

property Items.ItemObject (Item as HITEM) as Variant

Retrieves the item's ActiveX object, if the item was previously created by InsertControlItem property, or the original object being used when calling the InsertObjectItem property.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that was previously created by InsertControlItem or InsertObjectItem property.
Variant	An object that represents the ActiveX hosted by the item

The ItemObject retrieves the ActiveX object being created by the InsertControlItem method, or the object being hosted when using the InsertObjectItem property. Use the ItemControlID property to retrieve the control's identifier. Use the ItemHeight property to specify the item's height. If the item hosts an ActiveX control, the ItemHeight property specifies the height of the ActiveX control also.

property Items.ItemParent (Item as HITEM) as HITEM

Returns the handle of the item's parent item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the item's handle that indicates the parent item.

Use the ItemParent property to retrieve the parent item. Use the InsertItem property to insert child items. Use the InsertControlItem property to insert ActiveX controls. The SetParent method changes the item's parent at runtime. To verify if an item can be parent for another item you can call AcceptSetParent property. If the item has no parent the ItemParent property retrieves 0. If the ItemParent gets 0 for an item, than the item is called root. The control is able to handle more root items. To get the collection of root items you can use RootCount and RootItem properties. Use the ItemChild property to retrieve the first child item.

property Items.ItemPosition(Item as HITEM) as Long

Retrieves or sets a value that indicates the item's position in the children list.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long expression that indicates the item's position in the children list.

The ItemPosition property gets the item's position in the children items list. When the control sorts a column the position for each item can be changed. Use the handle of the item to identify an item. Use the SortChildren method to sort the child items. Use the SortOrder property to sort a column. Use the NextVisibleItem property to enumerate items as they are displayed. The SortableItem property specifies whether the item keeps its position after sorting.

property Items.ItemStrikeOut(Item as HITEM) as Boolean

Retrieves or sets the StrikeOut property of the Font object used to paint the item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the item uses strikeout font attribute to paint it.

If the ItemStrikeOut property is True, the cell's font is displayed with a horizontal line through it. Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.ItemToIndex (Item as HITEM) as Long

Retrieves the index of an item in the Items collection, given its handle.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long expression that indicates the index of Item in Items collection.

Use the ItemToIndex property to get the item's index in the Items collection. Use ItemPosition property to change the item's position. Use the ItemByIndex property to get an item giving its index. The ItemCount property counts the items in the control. The ChildCount property counts the child items.

property Items.ItemUnderline(Item as HITEM) as Boolean

Retrieves or sets the Underline property of the Font object used to paint the item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates if the item is underlined or not. True if the item is underlined, False, if the item is not underlined.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

property Items.ItemWidth(Item as HITEM) as Long

Retrieves or sets a value that indicates the item's width while it contains an ActiveX control.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that was previously created using InsertControlItem property.
Long	A long expression that indicates the item's width.

By default, the ItemWidth property is -1. If the ItemWidth property is -1, the control resizes the ActiveX control to fit the control's client area. Use the ItemHeight property to specify the item's height. The property has effect only if the item contains an ActiveX control. Use the InsertControlItem property to insert ActiveX controls. Use the ItemObject property to retrieve the ActiveX object that's hosted by an item. Use the CellWidth property to specify the width of the cell, when it contains inner cells. Use the SplitCell property to split a cell.

The ItemWidth property is interpreted like follows:

- If the ItemWidth property is greater than zero, the ItemWidth property indicates the width in pixels of the ActiveX control. The TreeColumnIndex property indicates the column where the ActiveX control is shown. For instance, ItemWidth = 64, indicates that the width of the inside ActiveX control is 64 pixels.
- If the ItemWidth property is zero, the ActiveX control uses the full item area to display the inside ActiveX control.
- If the ItemWidth property is -1, the TreeColumnIndex property indicates the column where the ActiveX control is shown and the inside ActiveX control is shown to the end of the control.
- If the ItemWidth property is less than -32000, the formula -(ItemWidth+32000) indicates the index of the column where the inside ActiveX is displayed. For instance, -32000 indicates that the cell in the first column displays the inside ActiveX control, -32001 indicates that the cell in the second column displays the inside ActiveX control, -32002 indicates that the cell in the third column displays the inside ActiveX control, and so on.
- If the ItemWidth property is -InnerCell or ItemCell, the ItemWidth property indicates the handle of the cell that shows the inside ActiveX. This option should be used when you need to display the ActiveX control in an inner cell. Use the SplitCell property to create inner cells, to divide a cell or to split a cell. For instance, .ItemWidth(.FirstVisibleItem) = -.InnerCell(.FirstVisibleItem, 1, 1) indicates that the inside ActiveX control is shown in the second inner cell in the second column, in the first visible item. Use the CellWidth property to specify the width of the inner cell.

property Items.ItemWindowHost (Item as HITEM) as Long

Retrieves the window's handle that hosts an ActiveX control when the item was created using InsertControlItem property.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that was previously created by InsertControlItem property.
Long	A long value that indicates the window handle that hosts the item's ActiveX.

The ItemWindowHost property retrieves the handle of the window that's the container for the item's ActiveX control. Use the InserControlItem method to insert an ActiveX control. Use the ItemObject property to access the ActiveX properties and methods. Use the hWnd property to get the handle of the control's window. 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.

property Items.ItemWindowHostCreateStyle(Item as HITEM) as Long

Retrieves or sets a value that indicates a combination of window styles used to create the ActiveX window host.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that was previously created by InsertControlItem property.
Long	A long value that indicates the container window's style.

The ItemWindowHostCreateStyle property specifies the window styles of the ActiveX's container window, when a new ActiveX control is inserted using the InsertControlItem method. The ItemWindowHostCreateStyle property has no effect for non ActiveX items. The ItemWindowHostCreateStyle property must be called during the ViewItemUpdate(exAddItem) event, like in the following samples. Generally, the ItemWindowHostCreateStyle property is useful to include WS_HSCROLL and WS_VSCROLL styles for a IWebBrowser control (WWW browser control), to include scrollbars in the browsed web page.

property Items.LastVisibleItem ([Partially as Variant]) as HITEM

Retrieves the handle of the last visible item.

Type	Description
Partially as Variant	A boolean expression that indicates whether the item is partially visible. By default, the Partially parameter is False.
HITEM	A long expression that indicates the item's handle that indicates the last visible item.

The LastVisibleItem property retrieves the handle for the last visible item. To get the first visible item use FirstVisibleItem property. Use the FirstVisibleItem, NextVisibleItem and IsItemVisible properties to get the items that fit the client area. Use the NextVisibleItem property to get the next visible item. Use the IsVisibleItem property to check whether an item fits the control's client area. The LastVisibleItem(False) property gets the handle of the last visible item that's not a partial item. The LastVisibleItem(True) property gets the handle of the last visible item no matter if it is partially visible or not.

property Items.LockedItem (Alignment as VAlignmentEnum, Index as Long) as HITEM

Retrieves the handle of the locked item.

Type	Description
Alignment as VAlignmentEnum	A VAlignmentEnum expression that indicates whether the locked item requested is on the top or bottom side of the control.
Index as Long	A long expression that indicates the position of item being requested.
HITEM	A long expression that indicates the handle of the locked item

property Items.LockedItemCount(Alignment as VAlignmentEnum) as Long

Specifies the number of items fixed on the top or bottom side of the control.

Type	Description
Alignment as VAlignmentEnum	A VAlignmentEnum expression that specifies the top or bottom side of the control.
Long	A long expression that indicates the number of items locked to the top or bottom side of the control.

A locked or fixed item is always displayed on the top or bottom side of the control no matter if the control's list is scrolled up or down. Use the LockedItemCount property to add or remove items fixed/locked to the top or bottom side of the control. Use the LockedItem property to access a locked item by its position. Use the ShowLockedItems property to show or hide the locked items. Use the CellValue property to specify the caption for a cell. Use the CountLockedColumns property to lock or unlock columns in the control. Use the ItemBackColor property to specify the item's background color. Use the ItemDivider property to merge the cells in the same item. Use the MergeCells method to combine one or more cells in a single cell.

property Items.MatchItemCount as Long

Retrieves the number of items that match the filter.

Type	Description
Long	A long expression that specifies the number of matching items in the control. The value could be a positive value if no filter is applied, or negative while filter is on.

The MatchItemCount property counts the number of items that matches the current filter criteria. At runtime, the MatchItemCount property 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 (ItemCount property). 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.

The MatchItemCount property returns a value as explained bellow:

- 0, the control displays/contains no items, and no filter is applied to any column
- -1, the control displays no items, and there is a filter applied (no match found)
- positive number, indicates the number of items within the control (ItemCount property)
- negative number, the absolute value minus 1, indicates the number of items that matches the current filter (match found)

method Items.MergeCells ([Cell1 as Variant], [Cell2 as Variant], [Options as Variant])

Merges a list of cells.

Type	Description
Cell1 as Variant	A long expression that indicates the handle of the cell being merged, or a safe array that holds a collection of handles for the cells being merged. Use the ItemCell property to retrieves the handle of the cell. The **first cell (in the list, if exists)** specifies the cell being displayed in the new larger cell.
Cell2 as Variant	A long expression that indicates the handle of the cell being merged, or a safe array that holds a collection of handles for the cells being merged. Use the ItemCell property to retrieves the handle of the cell. The first cell in the list specifies the cell being displayed in the new larger cell.
Options as Variant	Reserved.

The MergeCells method combines two or more cells into one cell. The data in the **first specified cell** is displayed in the new larger cell. All the other cells' data is not lost. Use the CellMerge property to merge or unmerge a cell with another cell in the same item. Use the ItemDivider property to display a single cell in the entire item. Use the UnmergeCells method to unmerge the merged cells. Use the CellValue property to specify the cell's value. Use the ItemCell property to retrieves the handle of the cell. Use the BeginMethod and EndUpdate methods to maintain performance, when merging multiple cells in the same time. The MergeCells methods creates a list of cells from Cell1 and Cell2 parameters that need to be merged, and the first cell in the list specifies the displayed cell in the merged cell. Use the SplitCell property to split a cell.

property Items.NextSiblingItem (Item as HITEM) as HITEM

Retrieves the next sibling of the item in the parent's child list.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the next sibling item's handle.

The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use ItemChild and NextSiblingItem properties to enumerate the collection of child items. Use the FirstVisibleItem property to get the first visible item in the control's client area. The NextVisibleItem property retrieves the handle of next visible item.

property Items.NextVisibleItem (Item as HITEM) as HITEM

Retrieves the handle of next visible item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the next visible item.

Use the NextVisibleItem property to access the visible items. The NextVisibleItem property retrieves 0 if there are no more visible items. Use the FirstVisibleItem property to get the first visible item in the control's client area. Use the RootItem property to get the first visible item in the list. The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use the IsItemVisible property to check whether an item fits the control's client area. Use the ItemPosition property to change the position of the item. Use the SortOrder property to sort a column.

property Items.PathSeparator as String

Returns or sets the delimiter character used for the path returned by the FullPath and FindPath properties.

Type	Description
String	A string expression that indicates the delimiter character used for the path returned by the FullPath and FindPath properties.

By default the PathSeparator is "\". The PathSeparator property is used by properties like FullPath and FindPath.

property Items.PrevSiblingItem (Item as HITEM) as HITEM

Retrieves the previous sibling of the item in the parent's child list.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the previous sibling item.

The PrevSiblingItem retrieves 0 if there are no more previous sibling items. The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use the FirstVisibleItem property to retrieve the first visible item. Use the ItemParent property to retrieve the parent of the item. Use the RootItem property to get the first visible item in the list.

property Items.PrevVisibleItem (Item as HITEM) as HITEM

Retrieves the handle of previous visible item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the previous visible item.

The PrevVisibleItem property retrieves 0 if there are no previous visible items. The NextVisibleItem property retrieves the next visible item. Use the FirstVisibleItem property to retrieve the first visible item. Use the ItemParent property to retrieve the parent of the item. Use the RootItem property to get the first visible item in the list.

method Items.RemoveAllItems ()

Removes all items from the control.

Type	Description

The RemoveAllItems method remove all items in the control. The Clear method of Columns object clears the Items collection too. Use the RemoveItem method to remove an item from the control.

method Items.RemoveItem (Item as HITEM)

Removes the given item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle being removed.

The RemoveItem method removes an item. The RemoveItem method does not remove the item, if it contains child items. Use the RemoveAllItems method to remove all items in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while removing the items. The RemoveItem method can't remove an item that's locked. Instead you can use the LockedItemCount property to add or remove locked items. Use the IsItemLocked property to check whether an item is locked. The RemoveSelection method removes the selected items (including the descendents).

method Items.RemoveSelection ()

Removes the selected items (including the descendents).

Type	Description

The RemoveSelection method removes the selected items (including the descendents). The RemoveItem method removes a specific item. The UnselectAll method unselects all items in the list.

property Items.RootCount as Long

Retrieves the number of root objects in the Items collection.

Type	Description
Long	A long value that indicates the count of root items into Items collection.

A root item is an item that has no parent (ItemParent() = 0). Use the RootItem property of the Items object to enumerates the root items. Use the AddItem to add root items to the control. Use the InsertItem method to insert child items.

property Items.RootItem ([Position as Long]) as HITEM

Retrieves the handle of the root item given its index in the root items collection.

Type	Description
Position as Long	A long value that indicates the index of the root item.
HITEM	A long expression that indicates the handle of the root item.

A root item is an item that has no parent (ItemParent() = 0). Use the RootCount property of to count the root items. Use the AddItem to add root items to the control. Use the InsertItem method to insert child items. Use the FirstVisibleItem property to get the first visible item in the control's client area. The NextVisibleItem property retrieves the handle of next visible item. The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use the RootItem property to get the first visible item in the list. If you need to enumerate the items as they are added, you may use the ItemByIndex property.

property Items.SelectableItem(Item as HITEM) as Boolean

Specifies whether the user can select the item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being selectable.
Boolean	A boolean expression that specifies whether the item is selectable.

By default, all items are selectable, excepts the locked items that are not selectable. A selectable item is an item that user can select using the keys or the mouse. The SelectableItem property specifies whether the user can select an item. The SelectableItem property doesn't change the item's appearance. The LockedItemCount property specifies the number of locked items to the top or bottom side of the control. Use the ItemDivider property to define a divider item. Use the ItemForeColor property to specify the item's foreground color. Use the ItemBackColor property to specify the item's background color. Use the ItemFont, ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to assign a different font to the item. Use the EnableItem property to disable an item. A disabled item looks grayed, but it is selectable. For instance, the user can't change the check box state in a disabled item. Use the SelectItem property to select an item. The ItemFromPoint property gets the item from point. For instance, if the user clicks a non selectable item the ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) event is not fired. A non selectable item is not focusable as well. It means that if the incremental searching is on, the non selectable items are ignored. Use the SelectCount property to get the number of selected items. Use the SelForeColor and SelBackColor properties to customize the colors for selected items. Use the ItemHeight property and SelectableItem property to hide an item. The SortableItem property specifies whether the item keeps its position after sorting.

method Items.SelectAll ()

Selects all items.

Type	Description

Use the SelectAll method to select all visible items in the tree. The SelectAll method has effect only if the SingleSel property is False, if the control supports multiple items selection. Use the UnselectAll method to unselect all items in the list. Use the SelectItem property to select or unselect a specified item. Use the SelectedItem property to retrieve a value that indicates whether the item is selected or unselected. Use the SelectCount property to retrieve the number of selected items. The SelectPos property selects/unselects items by position. The Selection property selects/unselects items by index.

property Items.SelectCount as Long

Counts the number of items that are selected in the control.

Type	Description
Long	A long expression that identifies the number of selected items.

The SelectCount property counts the selected items in the control. The control supports single or multiple selection. Use SingleSel property of the control to enable multiple selection. Use the SelectedItem property to retrieve the handle of the selected item(s). Use the SelBackColor property to indicate the background color for selected items. Use the SelForeColor property to specify the foreground color for selected items. The FocusItem property specifies the handle of the focused item. For instance, if the control supports single selection the FocusItem property retrieves the handle of the selected item too. Use the FullRowSelect property to specify how the user can select the cells or items using the mouse. Use the SelectItem property to programmatically select an item giving its handle. The control fires the ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) event when user changes the selection in the control. Use the SelectableItem property to specify whether the user can select an item.

property Items.SelectedItem ([Index as Long]) as HITEM

Retrieves the selected item's handle given its index in the selected items collection.

Type	Description
Index as Long	Identifies the index of the selected item into selected items collection. if it is missing, 0 is used.
HITEM	A long expression that indicates the handle of the selected item.

The SelectedItem property gets the handle of the items being selected. If the control supports multiple selection, you can use the SelectCount property to find out how many items are selected in the control. Use the SingleSel property to enable single or multiple selection. If the control supports single selection only a single item can be selected at runtime. Use the SingleSel property to specify whether the control supports single or multiple selection. If the control supports single selection, the FocusItem and SelectedItem property gets the handle of the selected/focused item, that's the same. Use the SelectItem property to specify whether an item is selected or not. The control fires the ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) event when user changes the selection in the control. Use the SelForeColor and SelBackColor properties to specify colors for selected items. Use the SelectableItem property to specify whether the user can select an item.

property Items.Selection as Variant

Selects items by index.

Type	Description
Variant	A long expression that indicates the index of item being selected, if the SingleSel property is True, or a safe array that holds a collection of index of items being selected, if the SingleSel property is False.

The Selection property selects/unselects items by index. Use the SelectItem property to select an item giving its handle. The ItemPosition property gives the relative position, or the position of the item in the child items collection. Use the SelectPos property to select items by position. The SelectPos property selects an item giving its general position.

The SingleSel property specifies whether the control supports single or multiple-selection. Based on the SingleSel property the Selection value is:

- of long type, if the SingleSel property is True (by default). For instance Selection = 0, indicates that the control selects the item with the index 0.
- a safe array of VARIANT, if the SingleSel property is False. For instance Selection = Array(0,1), indicates that the control selects the item with the index 0 and 1.

property Items.SelectItem(Item as HITEM) as Boolean

Selects or unselects a specific item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle being selected.
Boolean	A boolean expression that indicates the item's state. True if the item is selected, and False if the item is not selected.

Use the SelectItem property to programmatically select an item. The SelectItem property indicates whether an item is selected or not selected, giving its handle. Use the SelectedItem property to get the selected items, giving their indexes. The control fires ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) event when the user changes the selection. The SelectCount property counts the selected items in the control, when the control supports multiple selection. Use the SingleSel property to specify whether the control supports single or multiple selection. If the SingleSel property is True, the user can select a single item only. Use the FullRowSelect property to specify how the user can select the cells or items using the mouse. The FocusItem property specifies the handle of the focused item. The control can have only a single focused item. If the control supports single selection, the FocusItem property gets the selected item too. Use the EnsureVisibleItem property to ensure that an item is visible. Use the SelBackColor property to indicate the background color for selected items. Use the SelForeColor property to specify the foreground color for selected items. The SelectPos property selects/unselects items by position. The Selection property selects/unselects items by index.

property Items.SelectPos as Variant

Selects items by position.

Type	Description
Variant	A long expression that indicates the position of item being selected, if the SingleSel property is True, or a safe array that holds a collection of position of items being selected, if the SingleSel property is False.

Use the SelectPos property to select items by position. The SelectPos property selects an item giving its general position. Use the SelectItem property to select an item giving its handle. The ItemPosition property gives the relative position, or the position of the item in the child items collection. The Selection property selects/unselects items by index.

The SingleSel property specifies whether the control supports single or multiple-selection. Based on the SingleSel property the SelectPos value is:

- of long type, if the SingleSel property is True (by default). For instance SelectPos = 0, indicates that the control selects the item with the position 0 (first item).
- a safe array of VARIANT, if the SingleSel property is False. For instance SelectPos = Array(0,1), indicates that the control selects the item with the position 0 and 1.

method Items.SetParent (Item as HITEM, NewParent as HITEM)

Changes the parent of the given item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
NewParent as HITEM	A long expression that indicates the handle of the newly parent item.

Use the SetParent property to change the parent item at runtime. Use the InsertItem property to insert child items. Use the InsertControlItem property to insert ActiveX controls. Use AcceptSetParent property to verify if the the parent of an item can be changed. The following VB sample changes the parent item of the first item: View1.Items.SetParent View1.Items(0), View1.Items(1). Use the ItemParent property to retrieve the parent of the item.

property Items.SortableItem(Item as HITEM) as Boolean

Specifies whether the item is sortable.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being sortable.
Boolean	A boolean expression that specifies whether the item is sortable.

By default, all items are sortable. A sortable item can change its position after sorting. An unsortable item keeps its position after user performs a sort operation. Thought, the position of an unsortable item can be changed using the ItemPosition property. Use the SortableItem to specify a group item, a total item or a separator item. An unsortable item is not counted by a total field. The SortType property specifies the type of repositioning is being applied on the column when a sort operation is performed. The SortOrder property specifies whether the column is sorted ascendant or descendent. Use the SortChildren method to sort the items. Use the AllowSort property to avoid sorting a column when the user clicks the column. The ItemDivider property indicates whether the item displays a single cell, instead showing all cells. The SelectableItem property specifies whether an item can be selected.

The following screen shots shows the control when no column is sorted: (Group 1 and Group 2 has the SortableItem property on False)

Name	A	B	C
		Group 1	
Child 1	1	2	3
Child 2	4	5	6
		Group 2	
Child 1	1	2	3
Child 2	4	5	6

The following screen shots shows the control when the column A is being sorted: (Group 1 and Group 2 keeps their original position after sorting)

Name	A	▽ B	C
		Group 1	
Child 2	4	5	6
Child 1	1	2	3
		Group 2	
Child 2	4	5	6
Child 1	1	2	3

method Items.SortChildren (Item as HITEM, ColIndex as Variant, Ascending as Boolean)

Sorts the child items of the given parent item in the control.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption.
Ascending as Boolean	A boolean expression that defines the sort order. True means ascending, False means descending.

The SortChildren method will not recurse through the grid, only the immediate children of item will be sorted. After sort, the control ensures that the focused item fits the control's client area. Use the FocusItem property to retrieve the focused item. If your control acts like a simple list you can use the following line of code to sort ascending the list by first column: View1.Items.SortChildren 0, 0, True. To change the way how a column is sorted use SortType property of Column object. The SortChildren property doesn't display the sort icon on column's header. The control automatically sorts the children items when user clicks on column's header. Use the SortOnClick property to disable sorting columns by clicking in the columns header. Use the SortOrder property to get the column sorted, and to display the sorting icon in the column's header. The EnsureOnSort property prevents scrolling the control's content when the user sorts items. The SortableItem property specifies whether the item keeps its position after sorting. Use the AllowSort property to avoid sorting a column when the user clicks the column.

property Items.SplitCell ([Item as Variant], [ColIndex as Variant]) as Variant

Splits a cell, and returns the inner created cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item where a cell is being divided, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Variant	A long expression that indicates the handle of the cell being created.

The SplitCell method splits a cell in two cells. The newly created cell is called inner cell. The SplitCell method always returns the handle of the inner cell. If the cell is already divided using the SplitCell method, it returns the handle of the inner cell without creating a new inner cell. You can split an inner cell too, and so you can have a master cell divided in multiple cells. Use the CellWidth property to specify the width of the inner cell. Use the CellValue property to assign a value to a cell. Use the InnerCell property to access an inner cell giving its index. Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the owner of the cell. Use the UnsplitCell method to remove the inner cell if it exists. Use the MergeCells property to combine two or more cells in a single cell. Use the SelectableItem property to specify the user can select an item. Include the exIncludeInnerCells flag in the FilterList property and so the drop down filter window lists the inner cells too.

method Items.UnmergeCells ([Cell as Variant])

Unmerges a list of cells.

Type	Description
Cell as Variant	A long expression that indicates the handle of the cell being unmerged, or a safe array that holds a collection of handles for the cells being unmerged. Use the ItemCell property to retrieves the handle of the cell.

Use the UnmergeCells method to unmerge merged cells. Use the MergeCells method or CellMerge property to combine (merge) two or more cells in a single one. The UnmergeCells method unmerges all the cells that was merged. The CellMerge property unmerges only a single cell. The rest of merged cells remains combined.

method Items.UnselectAll ()

Unselects all items.

Type	Description

Use the UnselectAll method to unselect all items in the list. The UnselectAll method has effect only if the SingleSel property is False, if the control supports multiple items selection. Use the SelectAll method to select all items in the list. Use the SelectItem property to select or unselect a specified item. Use the SelectedItem property to retrieve a value that indicates whether the item is selected or unselected. Use the SelectCount property to retrieve the number of selected items. The SelectPos property selects/unselects items by position. The Selection property selects/unselects items by index. The RemoveSelection method removes the selected items (including the descendents).

method Items.UnsplitCell ([Item as Variant], [ColIndex as Variant])

Unsplits a cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.

Use the UnsplitCells method to remove the inner cells. The SplitCell method splits a cell in two cells, and retrieves the newly created cell. The UnsplitCell method has no effect if the cell contains no inner cells. The UnplitCells method remove recursively all inner cells. For instance, if a cell contains an inner cell, and this inner cell contains another inner cell, when calling the UnplitCells method for the master cell, all inner cells inside of the cell will be deleted. Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the owner of the cell. Use the InnerCell property to access an inner cell giving its index. Use the UnmergeCells method to unmerge merged cells. ("Merge" means multiple cells in a single cell, "Split" means multiple cells **inside** a single cell).

property Items.VisibleCount as Long

Retrieves the number of visible items.

Type	Description
Long	Counts the visible items.

Use FirstVisibleItem and NextVisibleItem properties to determine the items that fit the client area. Use the IsItemVisible property to check whether an item fits the control's client area. Use the ItemCount property to count the items in the control. Use the ChildCount property to count the child items

property Items.VisibleItemCount as Long

Retrieves the number of visible items.

Type	Description
Long	A long expression that specifies the number of visible items in the control. The value could be a positive value if no filter is applied, or negative while filter is on.

The VisibleItemCount property counts the number of visible items in the list. For instance, you can use the VisibleItemCount property to get the number the control displays once the user applies a filter.

The VisibleItemCount property returns a value as explained bellow:

- 0, the control displays/contains no items, and no filter is applied to any column
- -1, the control displays no items, and there is a filter applied (no match found)
- positive number, indicates the number of visible items, and the control has no filter applied to any column
- negative number, the absolute value munus 1, indicates the number of visible items, and there is a filter applied (match found)

The VisibleCount property retrieves the number of items being displayed in the control's client area. Use FirstVisibleItem and NextVisibleItem properties to determine the items being displayed in the control's client area. Use the IsItemVisible property to check whether an item fits the control's client area. Use the ItemCount property to count the items in the control. Use the ChildCount property to count the child items

View object

The View object supports the following properties and methods:

Name	Description
ActiveView	Gets the active view.
AllowGroupBy	Indicates whether the view supports Group-By view.
AllowSelectNothing	Specifies whether the current selection is erased, once the user clicks outside of the items section.
ApplyFilter	Applies the filter.
AutoDrag	Gets or sets a value that indicates the way the component supports the AutoDrag feature.
AutoSearch	Enables or disables the incremental searching feature.
BeginUpdate	Maintains performance when items are added to the view one at a time. This method prevents the view from painting until the EndUpdate method is called.
CheckImage	Retrieves or sets a value that indicates the image used by cells of checkbox type.
ChildView	Gets the child view (next).
ClearFilter	Clears the filter.
ColumnAutoResize	Returns or sets a value indicating whether the view will automatically size its visible columns to fit on the view's client width.
Columns	Retrieves the control's column collection.
ColumnsAllowSizing	Retrieves or sets a value that indicates whether a user can resize columns at run-time.
ConditionalFormats	Retrieves the conditional formatting collection.
ContinueColumnScroll	Retrieves or sets a value indicating whether the view will automatically scroll the visible columns by pixel or by column width.
CopyTo	Exports the view's view to an EMF file.
CountLockedColumns	Retrieves or sets a value indicating the number of locked columns. A locked column is not scrollable.
DataSource	Specifies the control's data as an array, XML, ADO or DAO.
	Retrieves or sets a value that indicates the default item

DefaultItemHeight	height.
DrawGridLines	Retrieves or sets a value that indicates whether the grid lines are visible or hidden.
EndUpdate	Resumes painting the view after painting is suspended by the BeginUpdate method.
EnsureOnSort	Specifies whether the view ensures that the focused item fits the view's client area, when the user sorts the items.
EnsureVisibleColumn	Scrolls the view's content to ensure that the column fits the client area.
ExpandOnDblClick	Specifies whether the item is expanded or collapsed if the user dbl clicks the item.
ExpandOnKeys	Specifies a value that indicates whether the view expands or collapses a node when user presses arrow keys.
ExpandOnSearch	Expands items automatically while user types characters to search for a specific item.
Export	Exports the view's data to a CSV format.
FilterBarCaption	Specifies the filter bar's caption.
FilterBarDropDownHeight	Specifies the height of the drop down filter window proportionally with the height of the view's list.
FilterBarHeight	Specifies the height of the view'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.
FirstView	Gets the first view.
FullRowSelect	Enables full-row selection in the view.

GetItems	Gets the collection of items into a safe array,
GridLineColor	Specifies the grid line color.
GridLineStyle	Specifies the style for gridlines in the list part of the view.
Group	Forces the view to do a regrouping of the columns.
HasButtons	Adds a button to the left side of each parent item. The user can click the button to expand or collapse the child items as an alternative to double-clicking the parent item.
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 view's header height.
HeaderSingleLine	Specifies whether the view resizes the columns header and wraps the captions in single or multiple lines.
HeaderVisible	Retrieves or sets a value that indicates whether the the grid's header is visible or hidden.
HideSelection	Returns a value that determines whether selected item appears highlighted when a control loses the focus.
hWnd	Retrieves the view's window handle.
Indent	Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
Index	Indicates the index of the view.
IsGrouping	Indicates whether the view is grouping the items.
Items	Retrieves the control's item collection.
ItemsAllowSizing	Retrieves or sets a value that indicates whether a user can resize items at run-time.
Key	Specifies the index or the caption of the column that determines the key of the view.
LastView	Gets the last view.
Level	Indicates the split level of the view.
LinesAtRoot	Link items at the root of the hierarchy.
MarkSearchColumn	Retrieves or sets a value that indicates whether the searching column is marked or unmarked

Name	Specifies the index or the caption of the column that determines the name of the view.
NextView	Gets the next view (child).
ParentView	Gets the parent view (previously).
PrevView	Gets the previously view (parent).
RadioImage	Retrieves or sets a value that indicates the image used by cells of radio type.
RemoveSelection	Removes the selected items (including the descendents)
RightToLeft	Indicates whether the component should draw right-to-left for RTL languages.
Scroll	Scrolls the view's content.
ScrollBars	Returns or sets a value that determines whether the view has horizontal and/or vertical scroll bars.
ScrollBySingleLine	Retrieves or sets a value that indicates whether the view scrolls the lines to the end. If you have at least a cell that has SingleLine false, you have to check the ScrollBySingleLine property..
ScrollPos	Specifies the vertical/horizontal scroll position.
SearchColumnIndex	Retrieves or sets a value indicating the column's index that is used for auto search feature.
SelBackMode	Retrieves or sets a value that indicates whether the selection is transparent or opaque.
Select	Selects the path
SelectColumnIndex	Retrieves or sets a value that indicates the index of the selected column, if the FullRowSelect property is False.
SelectOnRelease	Indicates whether the selection occurs when the user releases the mouse button.
ShowFocusRect	Retrieves or sets a value indicating whether the view draws a thin rectangle around the focused item.
ShowLockedItems	Retrieves or sets a value that indicates whether the locked/fixed items are visible or hidden.
SingleSel	Retrieves or sets a value that indicates whether the view supports single or multiple selection.
SingleSort	Returns or sets a value that indicates whether the view supports sorting by single or multiple columns.

SortBarCaption	Specifies the caption being displayed on the view's sort bar when the sort bar contains no columns.
SortBarColumnWidth	Specifies the maximum width a column can be in the view's sort bar.
SortBarHeight	Retrieves or sets a value that indicates the height of the view's sort bar.
SortBarVisible	Retrieves or sets a value that indicates whether control's sort bar is visible or hidden.
SortOnClick	Retrieves or sets a value that indicates whether the view sorts automatically the data when the user click on column's caption.
Tag	Specifies any extra data associated with the view.
TreeColumnIndex	Retrieves or sets a value that indicates the index of column where the hierarchy lines are displayed.
Ungroup	Ungroups the columns, if they have been previously grouped.
Value	Indicates the value of the single active item on the specified column.
ValueList	Returns the list of values for all selected / active items in the view, on the specified column, separated by comma.
Values	Returns a safe array with all values of selected / active items in the view, on the specified column.
View	Gets the view giving its index or tag.
Width	Specifies the width of the view.
WidthToFit	Specifies the width of the view.

property View.ActiveView as View

Gets the active view.

Type	Description
View	A View object that specifies the active view (the last view with any active items inside).

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs. The CreateView event is fired as soon as the control creates a new view. The Items property retrieves the view' items collection. The Columns property retrieves the view's columns collection.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

property View.AllowGroupBy as Boolean

Indicates whether the control supports Group-By view.

Type	Description
Boolean	A Boolean expression that specifies whether the user can group the items.

By default, the AllowGroupBy property is False. Set the AllowGroupBy property on True, to allow the user to group the items by dragging the column's header to control's sort bar. The SortBarVisible property specifies whether the control's sort bar is visible or hidden. If the control's sort bar is visible, the user can drag and drop columns to it, so the column get sorted and items grouped. The *ViewItemUpdate(exAddGroupItem)* event is fired when a new grouping items is added to the control's list. *You can use the ViewItemUpdate(exAddGroupItem) event, to add headers or footers during grouping, customize the aggregate formula to be displayed on different columns, while dropping a column to the sortbar.* The Column.AllowGroupBy property may be used to prevent grouping a specific column. The AllowSort property indicates whether the user can sort a column by clicking the column's header. The IsGrouping property specifies whether the control is grouping/ungrouping items. During grouping, the control keeps the items indentation, in other words, a child item will be a child after or before grouping. The ViewEndChanging(exLayoutChange) event is fired when the user changes the layout of the control, including dragging a column to the sort bar. The SortBarColumnsCount property indicates the number of the columns being grouped. The SortBarColumn property indicates the column being sorted giving its position in the sort bar. The Group/Ungroup method groups or ungroup the control's list. For instance, you can remove the grouping items, by calling the Ungroup method. The GroupByTotalField property determines the formula to be applied to the column when it gets grouped. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted.

property View.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.

method View.ApplyFilter ()

Applies the filter.

Type	Description

The ApplyFilter method updates the control's content once that user sets the filter using the Filter and FilterType properties. Use the ClearFilter method to clear the control's filter. Use the DisplayFilterButton property to show the filter drop down button in the column's caption. Use the FilterBarDropDownHeight property to specify the height of the drop down filter window. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the ShowFilter method to show programmatically the column's drop down filter window.

property View.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.

property View.AutoSearch as Boolean

Enables or disables the incremental searching feature.

Type	Description
Boolean	A boolean expression that indicates whether the auto search is enabled or disabled.

By default, the AutoSearch property is True. The auto-search feature is is commonly known as incremental search. An incremental search begins searching as soon as you type the first character of the search string. As you type in the search string, the control selects the item (and highlight the portion of the string that match where the string (as you have typed it so far) would be found. The control supports 'starts with' or 'contains' incremental search as described in the AutoSearch property of the Column object. Use the ASCIILower and ASCIIUpper properties to specify the set of lower and upper characters when auto search feature is enabled. Use the ExpandOnSearch property to expand items automatically while user types characters to search for a specific item. Use the MarkSearchColumn property to specify whether the control draws a rectangle around the searching column. The SearchColumnIndex property determines the index of the searching column.

method View.BeginUpdate ()

Maintains performance when items are added to the control one at a time.

Type	Description

The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Use BeginUpdate and EndUpdate statement each time when the control requires more changes. Using the BeginUpdate and EndUpdate methods increase the speed of changing the control properties by preventing it from painting during changing.

property View.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. Use the PartialCheck property to allow partial check feature within the column. The ImageSize property defines the size (width/height) of the control's check boxes.

property View.ChildView as View

Gets the child view (next).

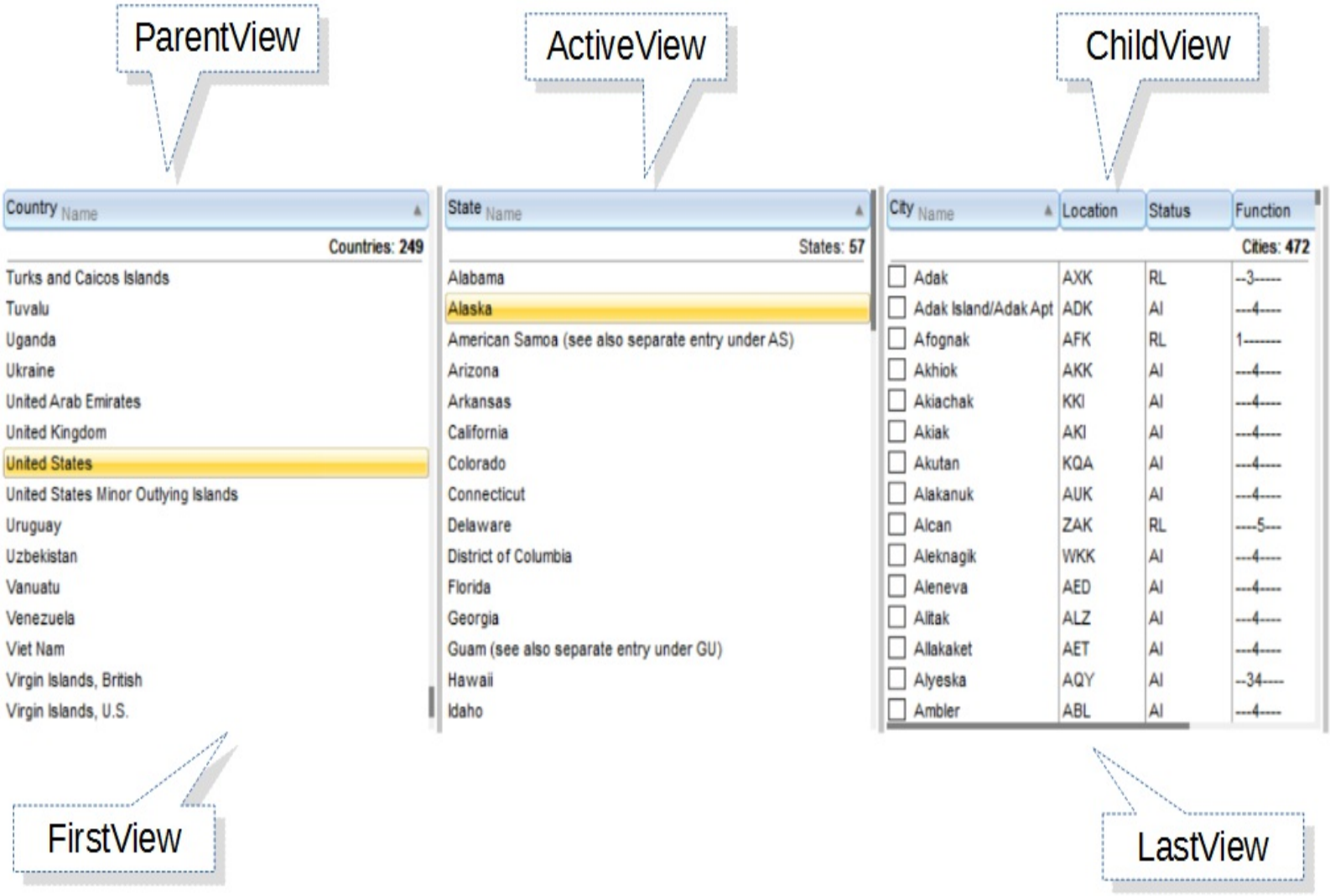
Type	Description
View	A View object that specifies the next / child view.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs.


ParentView
ActiveView
ChildView
Country Name
Countries: 249
Turks and Caicos Islands
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom
United States
United States Minor Outlying Islands
Uruguay
Uzbekistan
Vanuatu
Venezuela
Viet Nam
Virgin Islands, British
Virgin Islands, U.S.
State Name
States: 57
Alabama
Alaska
American Samoa (see also separate entry under AS)
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam (see also separate entry under GU)
Hawaii
Idaho
City Name
Location
Status
Function
Cities: 472
Adak AXK RL --3-----
Adak Island/Adak Apt ADK AI ---4----
Afognak AFK RL 1-------
Akhiok AKK AI ---4----
Akiachak KKI AI ---4----
Akiak AKI AI ---4----
Akutan KQA AI ---4----
Alakanuk AUK AI ---4----
Alcan ZAK RL ----5---
Aleknagik WKK AI ---4----
Aleneva AED AI ---4----
Alitak ALZ AI ---4----
Allakaket AET AI ---4----
Alyeska AQY AI --34----
Ambler ABL AI ---4----
FirstView
LastView

method View.ClearFilter ()

Clears the filter.

Type	Description

The method clears the Filter and FilterType properties for all columns in the control, excepts for exNumeric and exCheck values where only the Filter property is set on empty. The ApplyFilter method is automatically called when ClearFilter method is invoked. Use the FilterBarHeight property to hide the control's filter bar. Use the FilterBarCaption property to specify the caption in the control's filter bar. Use the Description property to change predefined strings in the control's filter bar. Use the ShowFilter method to show programmatically the column's drop down filter window.

property View.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 True. Use the ColumnAutoResize property to fit all visible columns in the control's client area. Use the Add method to add new columns to the control's Columns collection. Use the Width property to change the column's width. Use the Visible property to hide a column. Use the ContinueColumnScroll property to specify whether the user scrolls the control's content column by column or pixel by pixel. If the ColumnAutoResize property is True, the control does not display the control's horizontal scroll bar. Use the ScrollBars property to show or hide the control's scroll bars. By default, the control adds scroll bars when required.

property View.Columns as Columns

Retrieves the viewl's column collection.

Type	Description
Columns	A Columns object that indicates the viewl's column collection.

Use the Columns property to access the Columns collection. Use the Columns collection to add, remove or change columns. Use the Add method to add a new column to the control. Use the Items property to access the control's items collection. Use the AddItem, InsertItem, InsertControlItem method to add new items to the control. Use the DataSource property to add new columns and items to the control. Adding new items fails if the control has no columns.

property View.ColumnsAllowSizing as Boolean

Retrieves or sets a value that indicates whether a user can resize columns at run-time.

Type	Description
Boolean	A Boolean expression that indicates whether a user can resize columns at run-time.

By default, the ColumnsAllowSizing property is False. A column can be resized only if the AllowSizing property is True. Use the DrawGridLines property to show or hide the control's grid lines. Use the HeaderVisible property to show or hide the control's header bar. The HeaderAppearance property specifies the appearance of the column in the control's header bar.

property View.ConditionalFormats as ConditionalFormats

Retrieves the conditional formatting collection.

Type	Description
ConditionalFormats	A ConditionalFormats object that indicates the control's ConditionalFormats collection.

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. Use the Add method to format cells or items based on a formula. Use the Refresh method to refresh the control, if a change occurs in the conditional format collection. Use the CellValue property indicates the cell's caption or value.

The conditional format feature may change the cells and items as follows:

- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.
- Underline property. Underlines the cells or items
- Font property. Changes the font for cells or items.
- BackColor property. Changes the background color for cells or items, supports skins as well.
- ForeColor property. Changes the foreground color for cells or items.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to a column.

property View.ContinueColumnScroll as Boolean

Retrieves or sets a value indicating whether the control will automatically scroll the visible columns by pixel or by column width.

Type	Description
Boolean	A boolean expression indicating whether the control will automatically scroll the visible columns by pixel or by column width.

Use the ContinueColumnScroll property to define how the control scrolls the columns. Use the EnsureVisibleColumn method scrolls the control's content to ensure that the column fits the client area. Use the Scroll method to scroll the control's columns, column by column, if the ContinueColumnScroll property is False. Use the Visible property to hide a column. The ScrollBySingleLine property retrieves or sets a value that indicates whether the control scrolls the lines to the end, item by item. Use the ScrollBars property to hide the control's scroll bars.

property View.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. 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.

property View.CountLockedColumns as Long

Retrieves or sets a value indicating the number of locked columns.

Type	Description
Long	A long expression that indicates the number of locked columns.

The control is able to display two types of columns: locked and unlocked columns. A locked column is not scrollable, and it is fixed to the left side of the control. An unlocked control is scrollable. Use the CountLockedColumns property to define the number of columns that are in the locked area. Use the LockedItemCount property to lock or unlock items to the top or bottom side of the control. Use the MergeCells method to combine one or more cells in a single cell.

property View.DataSource as Variant

Specifies the control's data as an array, XML, ADO or DAO.

Type	Description
Variant	A VARIANT expression that could be a string, an object as explained bellow.

The control can automatically handle Array, XML, ADO, DAO, DataSet through the DataSource properties (control and view objects). You can specify the data source for the entire control through the DataSource property, or for a particular view using View.DataSource property. If an internal error occurs while using the DataSource property the Error event occurs. You can use the control's DataSource property to assign a data source for all views.

For instance,

- "...\sample.xml" opens the sample.xml file
- "...\sample.dbf" opens the specified sample.dbf table
- "Data Member=SELECT * FROM Orders ; Data Source=...\sample.accdb", opens the Orders table of the specified sample.accdb database
- "Data Member=SELECT * FROM Orders ; Data Source=...\sample.mdb", opens the Orders table of the specified sample.mdb database
- "Data Member=Orders ; Driver={Microsoft Access Driver (*.mdb)} ; DBQ=...\sample.mdb", opens the Orders table of sample.mdb database, using ODBC
- "Data Member=Orders ; Driver={Microsoft Access Driver (*.mdb)} ; DBQ=...\sample.mdb", opens the Orders table of sample.mdb database, using ODBC
- "Data Member=SELECT * FROM [Sheet1$] ; Driver={Microsoft Excel Driver (*.xls)} ; DBQ=...\sample.xls ; DriverID=790" reads the Sheet1 worksheet of the sample.xml file (Excel)

where ... indicates the full path to the sample file.

property View.DefaultItemHeight as Long

Retrieves or sets a value that indicates the default item height.

Type	Description
Long	A long expression that indicates the default item's height.

The DefaultItemHeight property specifies the height of the items. Changing the property fails if the control contains already items. You can change the DefaultItemHeight property at design time, or at runtime, before adding any new items to the Items collection. Use the ItemHeight property to specify the height of a specified item. Use the ScrollBySingleLine property when using the items with different heights. Use the CellSingleLine property to specify whether the cell displays the caption using multiple lines.

property View.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 indicates whether the grid lines are visible or hidden.

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.

method View.EndUpdate ()

Resumes painting the control after painting is suspended by the BeginUpdate method.

Type	Description

Use BeginUpdate and EndUpdate statement each time when the control requires more changes. Using the BeginUpdate and EndUpdate methods increase the speed of changing the control properties by preventing it from painting during changing.

property View.EnsureOnSort as Boolean

Specifies whether the control ensures that the focused item fits the control's client area, when the user sorts the items.

Type	Description
Boolean	A boolean expression that indicates whether the control ensures that the focused item fits the control's client area after sorting the items.

By default, the EnsureOnSort property is True. If the EnsureOnSort property is True, the control calls the EnsureVisibleItem method to ensure that the focused item (FocusItem property) fits the control's client area, once items get sorted. Use the SortOrder property to sort a column. The SortChildren method sorts child items of an item. The EnsureOnSort property prevents scrolling of the control when child items are sorted.

method View.EnsureVisibleColumn (Column as Variant)

Scrolls the control's content to ensure that the column fits the client area.

Type	Description
Column as Variant	A long expression that indicates the column's index being scrolled, or a string expression that indicates the column's caption or the column's key.

This method ensures that a column is at least partially visible. The control scrolls the content if necessary. The control automatically calls EnsureVisibleColumn method when the user clicks a cell in the column. Use the EnsureVisibleItem method to ensure that a specified item fits the control's client area. Use the ScrollBars property to hide the control's scroll bars. Use the Scroll method to programmatically scroll the control's content.

property View.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.

Use the ExpandOnDblClick property to disable expanding or collapsing items when user dbl clicks an item. Use the ExpandOnKeys property to specify whether the control expands or collapses a node when user presses arrow keys. The ExpandOnSearch property specifies whether the control expands nodes when incremental searching is on (AutoSearch property is different than 0) and user types characters when the control has the focus. The control fires the DblClick event when user double clicks the control. Use the ExpandItem property to programmatically expand or collapse an item. In CardView mode, the ExpandOnDblClick property specifies whether a card is expanded or collapsed when a card is double clicked.

property View.ExpandOnKeys as Boolean

Specifies a value that indicates whether the control expands or collapses a node when user presses arrow keys.

Type	Description
Boolean	A boolean expression that indicates whether the control expands or collapses a node when user presses arrow keys.

Use the ExpandOnKeys property to specify whether the control expands or collapses a node when user presses arrow keys. By default, the ExpandOnKeys property is True. Use the ExpandOnDblClick property to specify whether the control expands or collapses a node when user dbl clicks a node. The ExpandOnSearch property specifies whether the control expands nodes when incremental searching is on (AutoSearch property is different than 0) and user types characters when the control has the focus. If the ExpandOnKeys property is False, the user can't expand or collapse the items using the + or - keys on the numeric keypad. Use the ExpandItem property to programmatically expand or collapse an item. In CardView mode, the ExpandOnKeys property allows expanding or collapsing the cards using the + or - keys on the numeric keypad.

property View.ExpandOnSearch as Boolean

Expands items automatically while user types characters to search for a specific item.

Type	Description
Boolean	A boolean expression that indicates whether the control expands items while user types characters to search for items.

Use the ExpandOnSearch property to expand items while user types characters to search for items using incremental search feature. By default, the ExpandOnSearch property is False. Use the AutoSearch property to enable or disable incremental searching feature. Use the AutoSearch property of the Column object to specify the type of incremental searching being used within the column. The ExpandOnSearch property has no effect when the AutoSearch property is False. For instance, if the ExpandOnSearch property is True, the control fires the ViewItemStateStartChanging(exExpandItem) event for items that have the ItemHasChildren property is True, when user types characters.

method View.Export ([Destination as Variant], [Options as Variant])

Exports the control's data to a CSV format.

Type	Description
Destination as Variant	A String expression that specifies the file to be created, or empty, so the Export method returns the result as a string.
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.

The Export method exports the control's DATA to a CSV format. The Export method can export a collection of columns from selected, visible, check or all items.

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. The CellState property indicates the state of the cell's checkbox.
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].
4. the forth field could be **ansi** or **unicode**, which indicates the format to save the control's content to Destination. If missing, the control's configuration specifies the way the control's content is serialized, such as the ANSI version saves as ANSI, while the UNICODE version will export the control's data in UNICODE format. The Version property specifies the version of the control, and if this includes the UNICODE string, it indicates that you are running the UNICODE version of the control. For instance, Export(Destination,"|||ansi") saves the control's content to destination in ANSI format.

The Destination parameter indicates the file to be created where exported date should be

saved. For instance, Export(Destination,"sel|0,1|;") exports the cells from columns 0, 1 from the selected items, to a CSV using the ; character as a field separator. If Destination is empty or missing, the Export returns the result as a string.

The "CSV" refers to any file that:

- 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

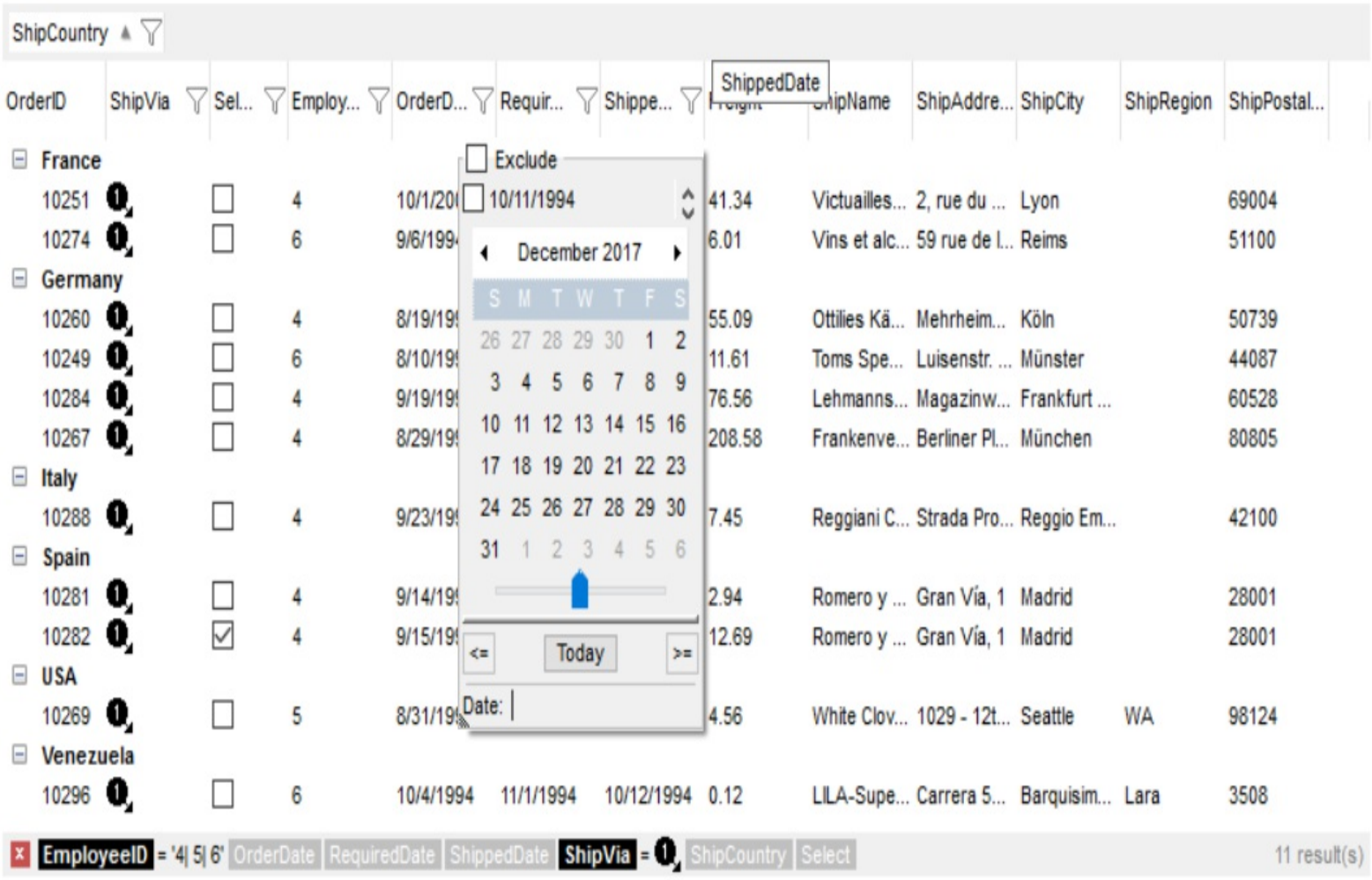
You can use the CopyTo to export the control's view to clipboard/EMF file.

property View.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.

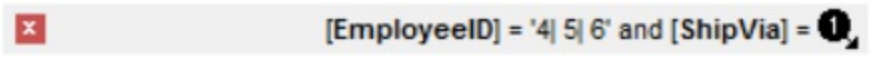


For instance:

- "no filter", shows no filter caption all the time

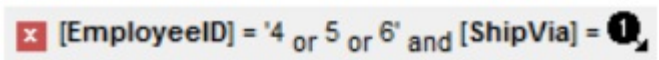
- "" displays no filter bar, if no filter is applied, else it displays the current filter

- "`<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



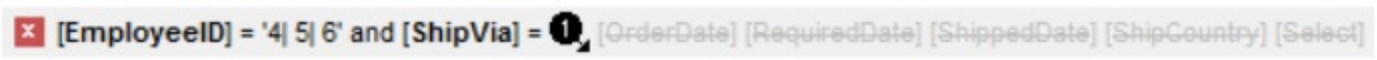
- "value replace ` and ` with `<fgcolor=FF0000> and </fgcolor>`", replace the AND keyword with a different foreground color

- "value replace ` and ` with `<off 4> and </off>` replace `|` with ` <off 4>or</off> ` replace ` ` with ` `", replaces the AND and | values

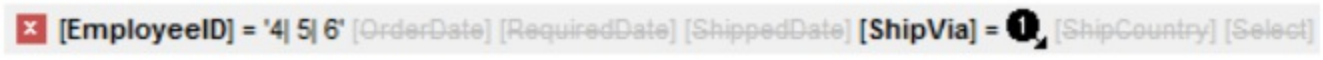


- "value replace `[` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `]` with ` </b></bgcolor></fgcolor>`", highlights the columns being filtered with a different background/foreground colors.

- "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. The VisibleItemCount property specifies the number of visible items in the list. The MatchItemCount property returns the number of matching items. 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 as indicated by ItemCount property. 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 as indicated by VisibleItemCount property. 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 as indicated by MatchItemCount property. 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 (ItemCount property). 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. A column is available to be filtered, if the DisplayFilterButton property of the Column object, is True. 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. A column is available to be filtered, if the DisplayFilterButton property of the Column object, is True. 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) no matter if the DisplayFilterButton property is True or False. 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 $_{subscript}$ 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:

- **<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 View.FilterBarDropDownHeight as Double

Specifies the height of the drop down filter window proportionally with the height of the control's list.

Type	Description
Double	A double expression that indicates the height of the drop down filter window.

Use the FilterBarDropDownHeight property to specify the height of the drop down window filter window. By default, the FilterBarDropDownHeight property is 0.5. It means, the height of the drop down filter window is half of the height of the control's list. Use the FilterBarDropDownWidth property to specify the width of the drop down filter window. Use the DisplayFilterButton property to display a filter button to the column's caption. Use the ShowFilter method to show programmatically the column's drop down filter window.

property View.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 ShowFilter method to show programmatically the column's drop down filter window.

property View.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. The DisplayFilterButton property specifies whether the column's header displays a filter button. The VisibleItemCount property retrieves the number of visible items in the list. Use the FilterBarCaption property to change the caption in the filter bar once a new filter is applied. 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 $_{subscript}$ 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:

- **<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:

- FilterBarPromptVisible property is True
- FilterBarPromptPattern property is Empty.

property View.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. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar.

property View.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. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way.

property View.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. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. The DisplayFilterButton property specifies whether the column's header displays a filter button. The VisibleItemCount property retrieves the number of visible items in the list. 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 View.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. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. 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 DisplayFilterButton property specifies whether the column's header displays a filter button. The VisibleItemCount property retrieves the number of visible items in the list.

property View.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.

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.

Use the Filter and FilterType properties to define a filter for a column. The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property, in code! Use the DisplayFilterButton property to display a drop down button to filter by a column. Use the CustomFilter property to define you custom filters.

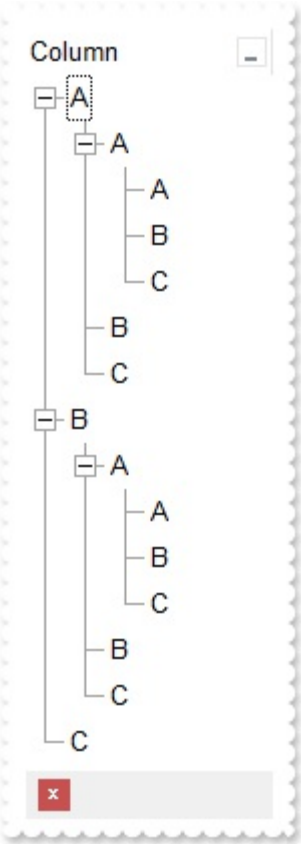
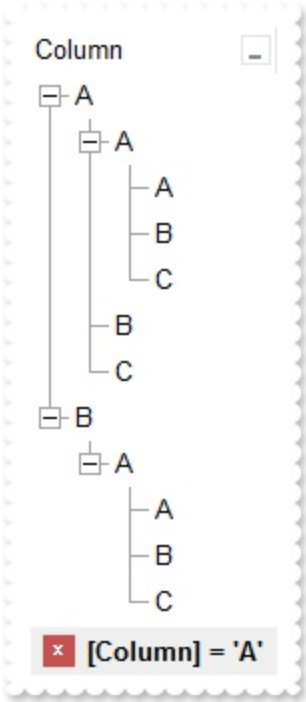
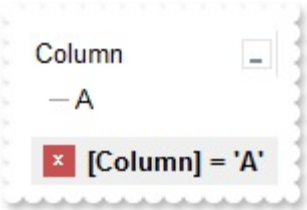
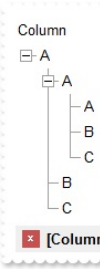
property View.FilterInclude as FilterIncludeEnum

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 Filter property and FilterType property to specify the column's filter. Use the ApplyFilter to apply the filter at runtime. 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 View.FirstView as View

Gets the first view.

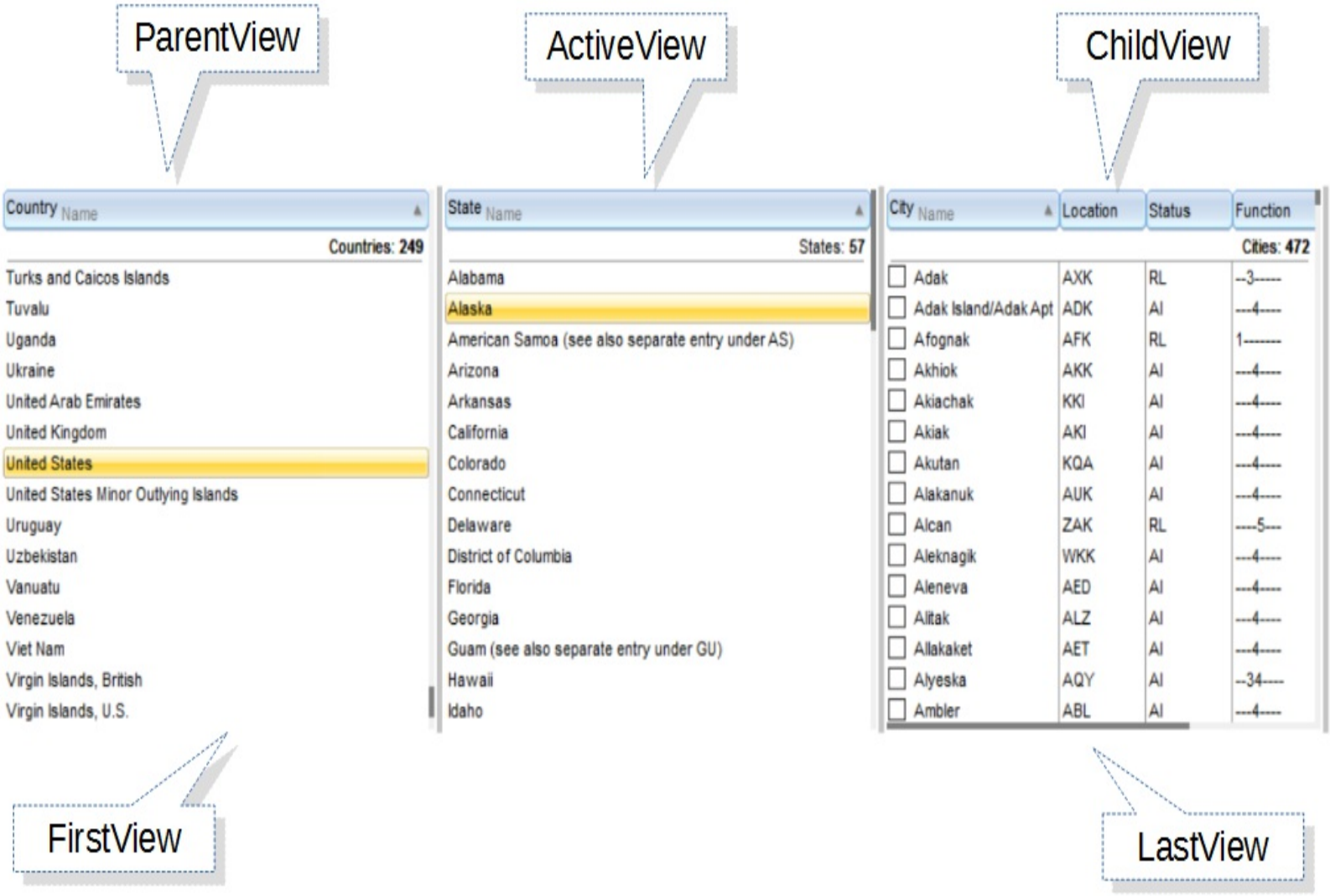
Type	Description
View	A View object that specifies the first view.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs.


ParentView
ActiveView
ChildView
Country Name
Countries: 249
Turks and Caicos Islands
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom
United States
United States Minor Outlying Islands
Uruguay
Uzbekistan
Vanuatu
Venezuela
Viet Nam
Virgin Islands, British
Virgin Islands, U.S.
State Name
States: 57
Alabama
Alaska
American Samoa (see also separate entry under AS)
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam (see also separate entry under GU)
Hawaii
Idaho
City Name
Location
Status
Function
Cities: 472
Adak AXK RL --3-----
Adak Island/Adak Apt ADK AI ---4----
Afognak AFK RL 1-------
Akhiok AKK AI ---4----
Akiachak KKI AI ---4----
Akiak AKI AI ---4----
Akutan KQA AI ---4----
Alakanuk AUK AI ---4----
Alcan ZAK RL ----5---
Aleknagik WKK AI ---4----
Aleneva AED AI ---4----
Alitak ALZ AI ---4----
Allakaket AET AI ---4----
Alyeska AQY AI --34----
Ambler ABL AI ---4----
FirstView
LastView

property View.FullRowSelect as CellSelectEnum

Enables full-row selection in the control.

Type	Description
CellSelectEnum	A CellSelectEnum expression that indicates whether the entire row is selected.

Use the FullRowSelect property to determine when the item or cell is selected. If the FullRowSelect property is exColumnSel, the SelectColumnIndex property determines the selected column. By default, the FullRowSelect property is exItemSel, and so the entire item is selected. If the FullRowSelect property is exRectSel property, the user can selects a range of cells by dragging. Use the Selected property to determine whether a cell is selected, if the FullRowSelect property is exRectSel. Use the SingleSel property to allow multiple items/cells in the selection. For instance, the FullRowSelect = True (boolean value) is the same as FullRowSelect = exItemSel, and FullRowSelect = False is the same as FullRowSelect = exColumnSel.

method View.GetItems (Options as Variant)

Gets the collection of items into a safe array,

Type	Description
Options as Variant	Specifies a long expression as follows: • if **0**, the result is a two-dimensional array with cell's values. The list includes the *collapsed* items, and the items are included as they are displayed (sorted, filtered). This option exports the values of cells. This option exports the values of the cells (CellValue property). • if **1**, the result the one-dimensional array of handles of items in the control as they are displayed (sorted, filtered). The list *does not include the collapsed items*. For instance, the first element in the array indicates the handle of the first item in the control, which can be different that FirstVisibleItem result, even if the control is vertically scrolled. This option exports the handles of the items. For instance, you can use the ItemToIndex property to get the index of the item based on its handle. • **else if other, and the number of columns is 1**, the result is a one-dimensional array that includes the items and its child items as they are displayed (sorted, filtered). In this case, the array may contains other arrays that specifies the child items. The list includes the *collapsed* items, and the items are included as they are displayed (sorted, filtered). This option exports the values of the cells (CellValue property) If missing, the Options parameter is 0. If the control displays no items, the result is an empty object (VT_EMPTY).

Return	Description
Variant	A safe array that holds the items in the control. If the control has a single column, the GetItems returns a single dimension array (object[]), else The safe array being returned has two dimensions (object[,]). The first

	dimension holds the collection of columns, and the second holds the cells.

The GetItems method to get a safe array that holds the items in the control. The GetItems method gets the items as they are displayed, sorted and filtered. Also, the GetItems method collect the child items as well, no matter if the parent item is collapsed. The method returns nothing if the control has no columns or items. Use the Items property to access the items collection. You can use the GetItems(1) method to get the list of handles for the items as they are displayed, sorted and filtered. The GetItems method returns an empty expression (VT_EMPTY), if there is no items in the result.

/NET Assembly:

The following C# sample converts the returned value to a object[] when the control contains a single column:

```
object[] Items = (object[])view1.GetItems()
```

or when the control contains multiple columns, the syntax is as follows:

```
object[,] Items = (object[,])view1.GetItems()
```

The following VB.NET sample converts the returned value to a Object() when the control contains a single column:

```
Dim Items As Object() = View1.GetItems()
```

or when the control contains multiple columns, the syntax is as follows:

```
Dim Items As Object(,) = View1.GetItems()
```

property View.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 View.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 exViewLinesDot. 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.

method View.Group ()

Forces the control to do a regrouping of the columns.

Type	Description

The Group method forces the control to re-group the items. The AllowGroupBy property specifies whether the control supports Group-By feature. The Group method has no effect if the AllowGroupBy property is False. The Ungroup method un-groups the items in the control's list. During execution any of these methods, the IsGrouping property returns True. You can call the SortOrder property to sort and group by specified column. Use the SortType property to determine the way how the column is sorted. The *ViewItemUpdate(exAddGroupItem)* event is fired when a new grouping items is added to the control's list. *You can use the ViewItemUpdate(exAddGroupItem) event, to add headers or footers during grouping, customize the aggregate formula to be displayed on different columns, while dropping a column to the sortbar.* The Column.AllowGroupBy property may be used to prevent grouping a specific column. The AllowSort property indicates whether the user can sort a column by clicking the column's header.

property View.HasButtons as ExpandButtonEnum

Adds a button to the left side of each parent item. The user can click the button to expand or collapse the child items as an alternative to double-clicking the parent item.

Type	Description
ExpandButtonEnum	An ExpandButtonEnum expression that indicates whether the control displays a + button to the left of each parent item.

The HasButtons property has effect only if the data is displayed as a grid. Use the InsertItem property to let the control displays your data as a grid. Use the TreeColumnIndex property to select the column where the hierarchy is displayed. Use the LinesAtRoot property to let the control displays a line that links the root items of the control. Use the CellVAlignment property to specify where the +/- AND the cell's caption is displayed in the item's client area. For instance, you can't have the +/- sign aligned to the top of the cell, and its caption aligned to the bottom. The +/- signs are always centered to the cell's caption, only the cell's caption can be aligned to the top or to the bottom of the cell's client area.

property View.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 InsertItem method to insert new items to the control. Use HasButtons property to hide the buttons displayed at the left of each parent item. Use the DrawGridLines property to display grid lines. The GridLineStyle property to specify the style for horizontal or/and vertical gridlines in the control. Use the InsertControlItem property to insert an ActiveX item.

property View.HeaderAppearance as AppearanceEnum

Retrieves or sets a value that indicates the header's appearance.

Type	Description
AppearanceEnum	A boolean expression that specifies the appearance of the columns header.

Use the HeaderAppearance property to define the appearance of the columns header bar. The user can't resize the columns at runtime, if the HeaderAppearance property is None2. Use the ColumnsAllowSizing property to allow resizing the columns, when the control's header bar is not visible. Use the Appearance property to define the control's appearance. Use the HeaderVisible property to hide the control's header bar.

property View.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. Use the LevelKey property to display the control's header bar using multiple levels. Use the FormatLevel property to display multiple levels in the column's header. If the control displays the header bar using multiple levels the HeaderHeight property gets the height in pixels of a single level in the header bar. The control's header displays multiple levels if there are two or more neighbor columns with the same non empty level key. Use the HTMLCaption property to display multiple lines in the column's caption. Use the Add method to add new columns to the control. *If the HeaderSingleLine property is False, the HeaderHeight property specifies the maximum height of the control's header when the user resizes the columns.*

property View.HeaderSingleLine as Boolean

Specifies whether the control resizes the columns header and wraps the captions in single or multiple lines.

Type	Description
Boolean	A boolean expression that specifies whether the header displays single or multiple lines.

By defauly, the HeaderSingleLine property is True. If the HeaderSingleLine property is False the control breaks the column's caption as soon as the user resizes the column. **In this case the HeaderHeight property specifies the maximum height of the control's header.** The initial height is computed based on the control's Font property. The Caption property specifies the caption of the column being displayed in the control's header. The HTMLCaption property specifies the HTML caption of the column being displayed in the column's header. Use the LevelKey property to display the control's header on multiple levels.

property View.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 Visible property to hide a particular column. Use the ColumnFromPoint property to access the column from point. If the control's header bar is hidden, the ColumnFromPoint property returns -1. Use the LevelKey property to allow multiple levels header bar. Use the FormatLevel property to display multiple levels in the column's header. 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. Use the AllowSizing property to disable resizing a column when user clicks the right margin of the column. Use the SortOnClick property to specify the action that control takes when the column's caption is clicked. Use the ColumnsAllowSizing property to allow resizing the columns, when the control's header bar is not visible. The Background(exCursorHoverColumn) property specifies the visual appearance of the column's header when the cursor hovers it.

property View.HideSelection as Boolean

Returns a value that determines whether selected item appears highlighted when a control loses the focus.

Type	Description
Boolean	A boolean expression that indicates whether the selected item appears highlighted when a control loses the focus.

By default, the HideSelection property is False. You can use this property to indicate which item is highlighted while another form, dialog box or control has the focus. Use the HideSelection property to hide the selected items when the control loses the focus. Use the SelBackColor property to indicate the background color for selected items. Use the SelForeColor property to specify the foreground color for selected items. Use the SelectItem property to select programmatically items. Use the SelectedItem and SelectCount property to retrieve the list of selected items. Use the SelectableItem property to specify whether an items can be selected.

property View.hWnd as Long

Retrieves the control's window handle.

Type	Description
Long	A long value that indicates the handle of the control's window.

Use the hWnd property to get the handle of the control's window. Use the ItemWindowHost property to get the handle of the container window that host an ActiveX control. 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.

property View.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

By default, the Indent property is 22 pixels. 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. Use the HasButtons property to define the +/- signs appearance. Use the TreeColumnIndex property to define the index of the column that displays the hierarchy. Use the InsertItem method to insert a child item. Use the InsertControlItem property to insert an ActiveX item.

property View.Index as Long

Indicates the index of the view.

Type	Description
Long	A long expression that specifies the index of the view (0-based)

The Index property specifies the index of the view on the control. The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs.

property View.IsGrouping as Boolean

Indicates whether the control is grouping the items.

Type	Description
Boolean	A Boolean expression that specifies whether the control is grouping or ungrouping the items.

The IsGrouping property determines whether the control is grouping/ungrouping the items. The AllowGroupBy property specifies whether the control supports Group-By feature. For instance, during grouping, the control may expand or collapse items, you can use the IsGrouping property to determine if the ViewItemStateStartChanging(exExpandItem)/ ViewItemStateEndChanging(exExpandItem) events occur due user interaction or control's grouping operation. The GroupItem property indicates the index of the column being grouped for specified grouping item. The Group/Ungroup method groups or ungroup the control's list. During execution any of these methods, the IsGrouping property returns True. The ViewEndChanging(exLayoutChange) event is fired when the user changes the layout of the control, including dragging a column to the sort bar. The SortBarColumnsCount property indicates the number of the columns being grouped. The SortBarColumn property indicates the column being sorted giving its position in the sort bar.

property View.Items as Items

Retrieves the view's item collection.

Type	Description
Items	Defines the viewl' Items collection.

Use the Items property to access the Items collection. Use the Items collection to add, remove or change the control items. Use the GetItems method to get the items collection into a safe array. Use the Columns property to access the control's Columns collection. Use the AddItem, InsertItem or InsertControlItem method to add new items to the control. Use the DataSource to add new columns and items to the control. Adding new items fails if the control has no columns.

property View.ItemsAllowSizing as ItemsAllowSizingEnum

Retrieves or sets a value that indicates whether a user can resize items at run-time.

Type	Description
ItemsAllowSizingEnum	An ItemsAllowSizingEnum expression that specifies whether the user can resize a single item at runtime, or all items, at once.

By default, the ItemsAllowSizing property is exNoSizing. Use the ItemsAllowSizing property to specify whether all items are resizable. Use the ItemAllowSizing property of the Items object to specify only when few items are resizable or not. Use the ItemHeight property to specify the height of the item. The CellSingleLine property specifies whether a cell displays its caption using multiple lines. The DefaultItemHeight property specifies the default height of the items, before loading data to your control .

property View.Key as Variant

Specifies the index or the caption of the column that determines the key of the view.

Type	Description
Variant	A long or string expression that specifies the index or the caption of the column that determines the key of the view.

The DataSource property can change the Key property using the Key field as explained bellow:

- **Key** or **Data Key**, specifies the index or the name of the field from the Data Member that generates keys for the current view. For instance, "**Key**=CountryCode" specifies that the CountryCode field of the current view generates keys for next child views. The Key property of the View object can be used to access the key of the view at runtime. If the Key refers to an exiting field/column in the current view, it means that the control generates the next view, once the user selects one or more items into the current view. If the Key is empty or points to an non-existing field/column in the current view, no view will be generated once an item in the current view is selected. The control fires the CreateView event once a new view requires to be created. The ViewStartChanging(exSelectChange) / ViewEndChanging(exSelectChange) event notifies your application once the selection into the view is changing. During any event, you can access the view that generated the event, using the View property of the control. The Select property of the control generates the path of the current selection for all views using the Key property of each View (separated by \ backslash character). For instance, the Select property could return "US\AK". The Key field is not required, and if missing no view will be generated once the user selects an item into the current view.

The Select property can select items using wild characters such as * or ?, if the view's SingleSel property is False.

property View.LastView as View

Gets the last view.

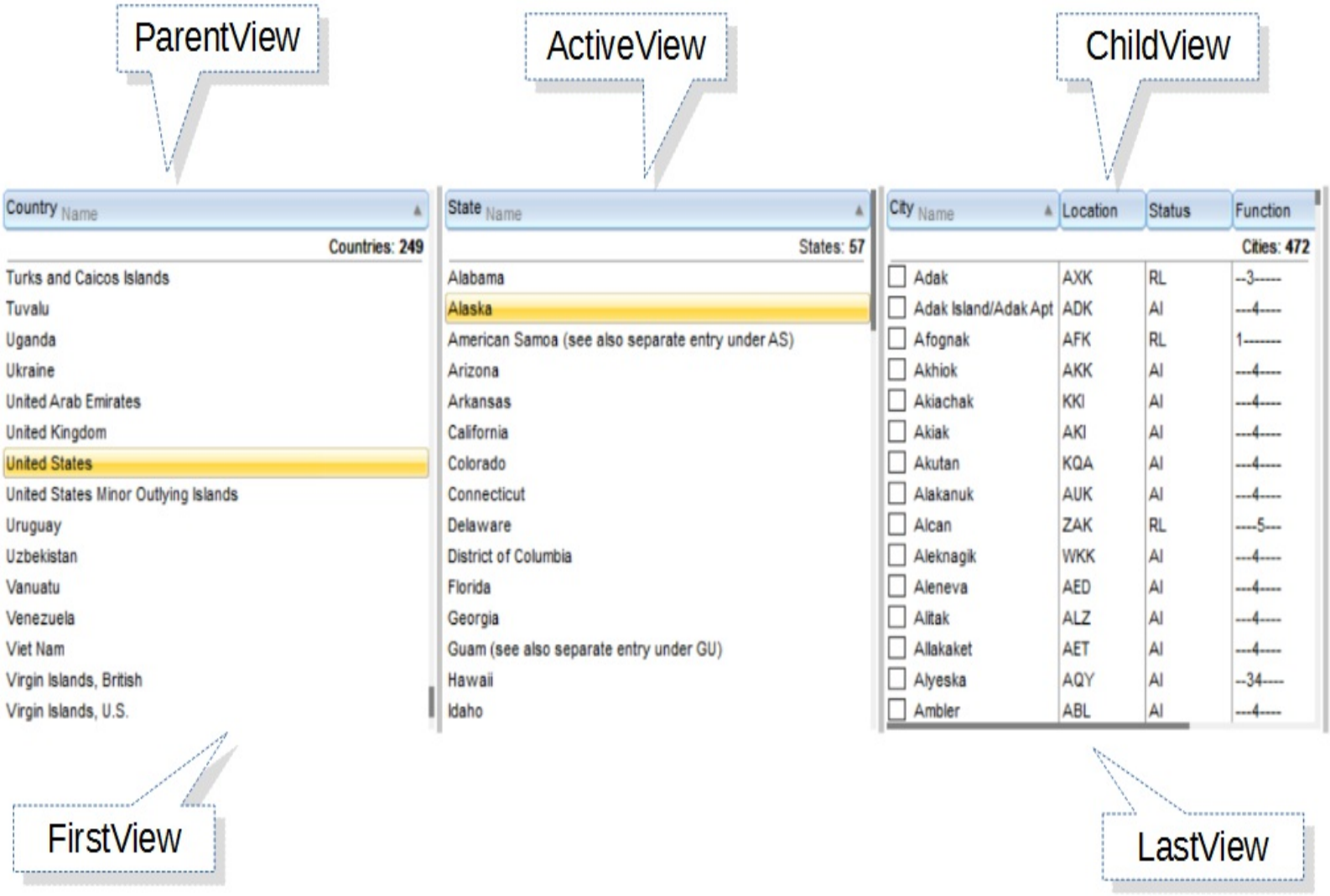
Type	Description
View	A View object that indicates the last view.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs.


ParentView
ActiveView
ChildView
Country Name
Countries: 249
Turks and Caicos Islands
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom
United States
United States Minor Outlying Islands
Uruguay
Uzbekistan
Vanuatu
Venezuela
Viet Nam
Virgin Islands, British
Virgin Islands, U.S.
State Name
States: 57
Alabama
Alaska
American Samoa (see also separate entry under AS)
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam (see also separate entry under GU)
Hawaii
Idaho
City Name
Location
Status
Function
Cities: 472
Adak AXK RL --3-----
Adak Island/Adak Apt ADK AI ---4----
Afognak AFK RL 1-------
Akhiok AKK AI ---4----
Akiachak KKI AI ---4----
Akiak AKI AI ---4----
Akutan KQA AI ---4----
Alakanuk AUK AI ---4----
Alcan ZAK RL ----5---
Aleknagik WKK AI ---4----
Aleneva AED AI ---4----
Alitak ALZ AI ---4----
Allakaket AET AI ---4----
Alyeska AQY AI --34----
Ambler ABL AI ---4----
FirstView
LastView

property View.Level as Long

Indicates the split level of the view.

Type	Description
Long	A Long expression that Indicates the split vertical level of the view.

The AllowSplitView property specifies whether the user can split the control into multiple-views. The SplitViewHeight property specifies the height of split panels, separated by comma. The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs. The ActiveView property gets the active view (the last view with any active items inside). The CreateView event is fired as soon as the control creates a new view. The Items property retrieves the view' items collection. The Columns property retrieves the view's columns collection.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

property View.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.

The control paints the hierarchy lines to the right if the Column's Alignment property is RightAlignment. The TreeColumnIndex property specifies the index of column where the hierarchy lines are painted. 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. Use the InsertItem method to insert a child item. Use the InsertControlItem property to insert an ActiveX item.

property View.MarkSearchColumn as Boolean

Retrieves or sets a value that indicates whether the searching column is marked or unmarked

Type	Description
Boolean	A boolean expression that indicates whether the searching column is marked or unmarked.

The control marks the searching column by drawing a rectangle around it. The SearchColumnIndex property determines the index of the searching column. Use the MarkSearchColumn property to hide the searching column. By default, the MarkSearchColumn property is True. The user can change the searching column by pressing the TAB ort Shift + TAB key. Use the AutoSearch property to specify whether the control enables the incremental searching feature. Use the AutoSearch property to specify the type of incremental searching the control supports within the column. Use the UseTabKey property to specify whether the control uses the TAB key.

property View.Name as Variant

Specifies the index or the caption of the column that determines the name of the view.

Type	Description
Variant	A long or string expression that specifies the index or the caption of the column that determines the name of the view.

The DataSource property can change the Name property using the Name field as explained bellow:

- **Name** or **Data Name**, indicates the index or the name of the field from the Data Member, that generates names for the Name property. For instance, "**Name**=CountryName", indicates that the CountryName column of the current view generate values for the Name property. The Name property of the control generates the path of the current selection for all views using the Name property of each View (separated by \ backslash character). For instance, the Name property could return "United States\Alaska\Anchorage", The Name field is not required. By default, the column with the index 0 specifies the name column.

The Name property of each View object specifies the index or the caption of the column that determines the name of the view. The Name property can select items using wild characters such as * or ?, if the view's SingleSel property is False. The Name property is similar with the Select property, excepts it uses the Name column to build the path.

property View.NextView as View

Gets the next view (child).

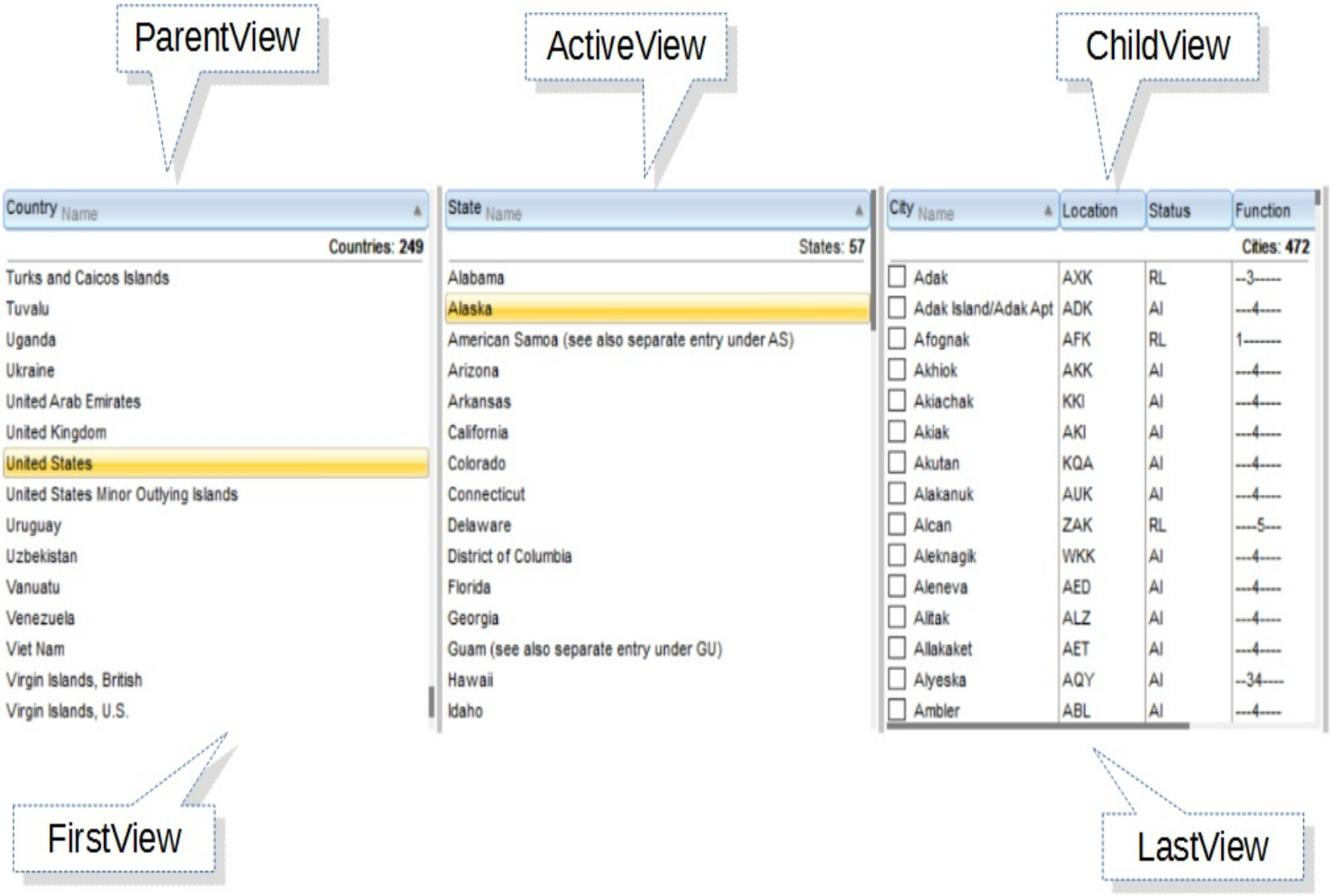
Type	Description
View	A View object that specifies the next / child view.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs.


ParentView
ActiveView
ChildView
Country Name
Countries: 249
Turks and Caicos Islands
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom
United States
United States Minor Outlying Islands
Uruguay
Uzbekistan
Vanuatu
Venezuela
Viet Nam
Virgin Islands, British
Virgin Islands, U.S.
State Name
States: 57
Alabama
Alaska
American Samoa (see also separate entry under AS)
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam (see also separate entry under GU)
Hawaii
Idaho
City Name
Location
Status
Function
Cities: 472
Adak AXK RL --3-----
Adak Island/Adak Apt ADK AI ---4----
Afognak AFK RL 1-------
Akhiok AKK AI ---4----
Akiachak KKI AI ---4----
Akiak AKI AI ---4----
Akutan KQA AI ---4----
Alakanuk AUK AI ---4----
Alcan ZAK RL ----5---
Aleknagik WKK AI ---4----
Aleneva AED AI ---4----
Alitak ALZ AI ---4----
Allakaket AET AI ---4----
Alyeska AQY AI --34----
Ambler ABL AI ---4----
FirstView
LastView

property View.ParentView as View

Gets the parent view (previously).

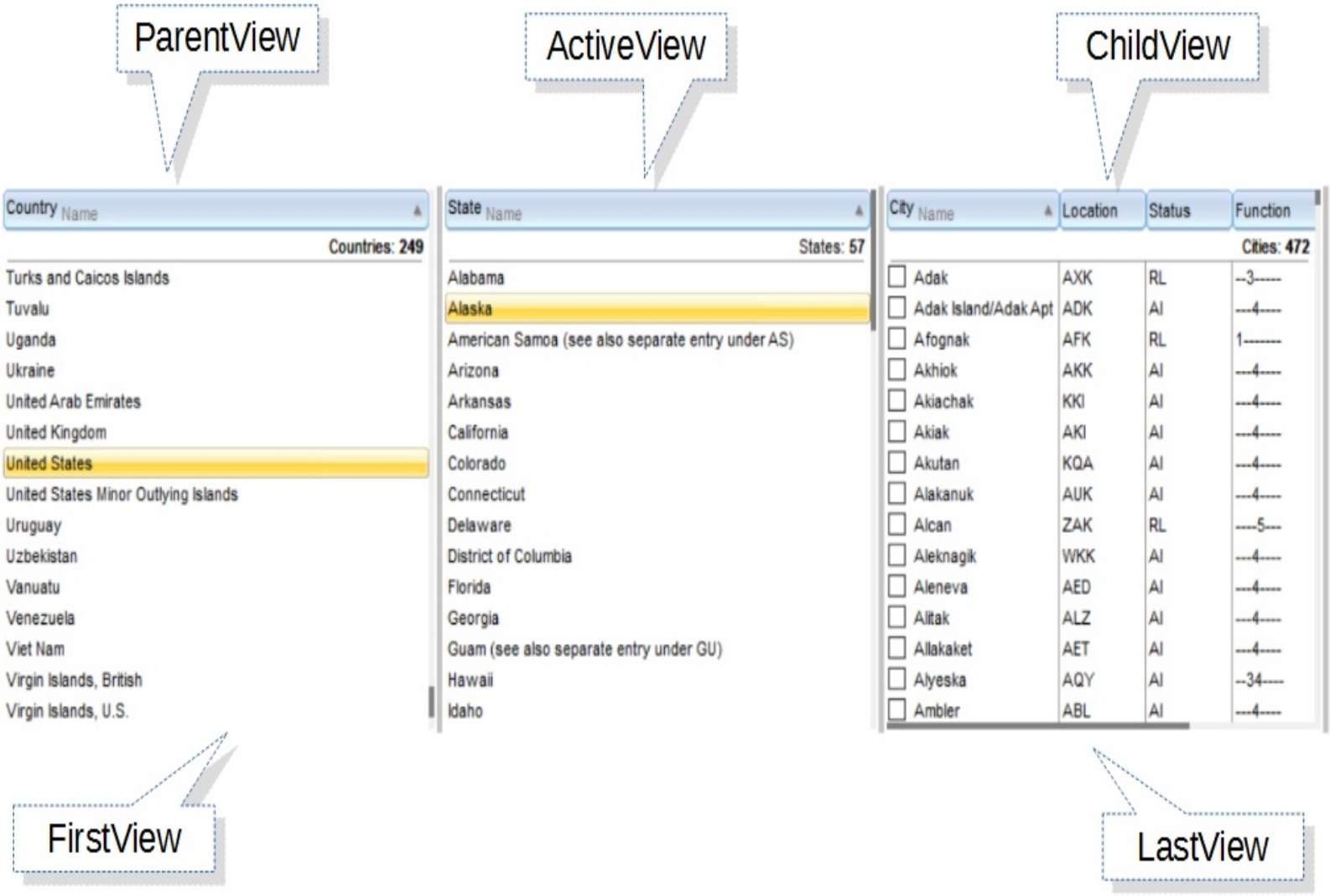
Type	Description
View	A View object that specifies the parent view.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs.


ParentView
ActiveView
ChildView
Country Name
Countries: 249
Turks and Caicos Islands
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom
United States
United States Minor Outlying Islands
Uruguay
Uzbekistan
Vanuatu
Venezuela
Viet Nam
Virgin Islands, British
Virgin Islands, U.S.
State Name
States: 57
Alabama
Alaska
American Samoa (see also separate entry under AS)
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam (see also separate entry under GU)
Hawaii
Idaho
City Name
Location
Status
Function
Cities: 472
Adak AXK RL --3-----
Adak Island/Adak Apt ADK AI ---4----
Afognak AFK RL 1-------
Akhiok AKK AI ---4----
Akiachak KKI AI ---4----
Akiak AKI AI ---4----
Akutan KQA AI ---4----
Alakanuk AUK AI ---4----
Alcan ZAK RL ----5---
Aleknagik WKK AI ---4----
Aleneva AED AI ---4----
Alitak ALZ AI ---4----
Allakaket AET AI ---4----
Alyeska AQY AI --34----
Ambler ABL AI ---4----
FirstView
LastView

property View.PrevView as View

Gets the previously view (parent).

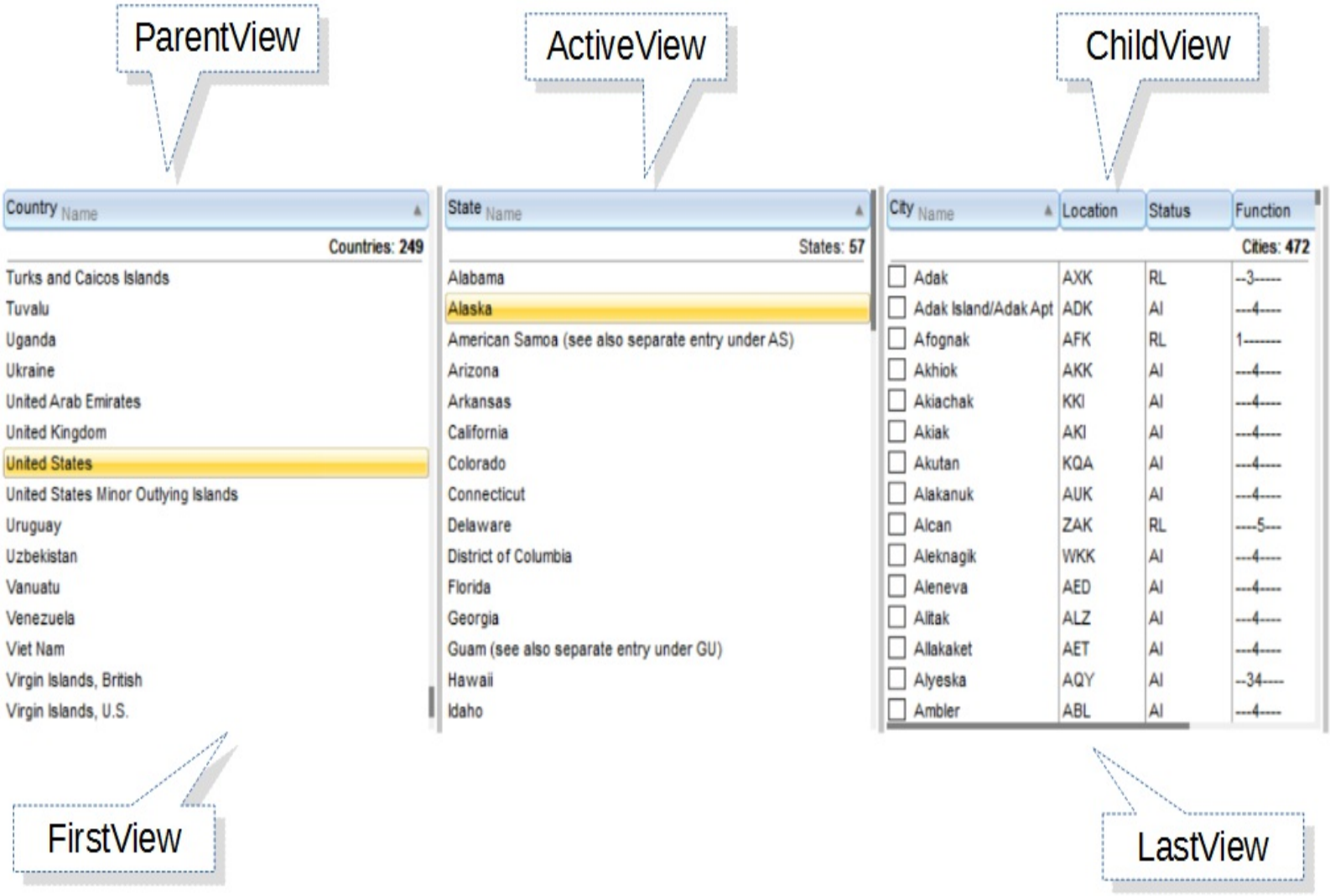
Type	Description
View	A View object that specifies the previously view (parent).

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

The ActiveView property gets the active view (the last view with any active items inside). The DefaultView property specifies the default view on the control. The View property returns the default view, in case it is not called during an event. During any event, the View property returns the view where the event occurs.


ParentView
ActiveView
ChildView
Country Name
Countries: 249
Turks and Caicos Islands
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom
United States
United States Minor Outlying Islands
Uruguay
Uzbekistan
Vanuatu
Venezuela
Viet Nam
Virgin Islands, British
Virgin Islands, U.S.
State Name
States: 57
Alabama
Alaska
American Samoa (see also separate entry under AS)
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam (see also separate entry under GU)
Hawaii
Idaho
City Name
Location
Status
Function
Cities: 472
Adak AXK RL --3-----
Adak Island/Adak Apt ADK AI ---4----
Afognak AFK RL 1-------
Akhiok AKK AI ---4----
Akiachak KKI AI ---4----
Akiak AKI AI ---4----
Akutan KQA AI ---4----
Alakanuk AUK AI ---4----
Alcan ZAK RL ----5---
Aleknagik WKK AI ---4----
Aleneva AED AI ---4----
Alitak ALZ AI ---4----
Allakaket AET AI ---4----
Alyeska AQY AI --34----
Ambler ABL AI ---4----
FirstView
LastView

property View.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 PartialCheck property to allow partial check feature within the column. Use the Images method to insert icons at runtime. The following samples require a control with icons, else nothing will be changed. The ImageSize property defines the size (width/height) of the control's radio buttons.

method View.RemoveSelection ()

Removes the selected items (including the descendents)

Type	Description

The RemoveSelection method removes the selected items (including the descendents). The RemoveItem method removes a specific item. The UnselectAll method unselects all items in the list.

property View.RightToLeft as Boolean

Indicates whether the component should draw right-to-left for RTL languages.

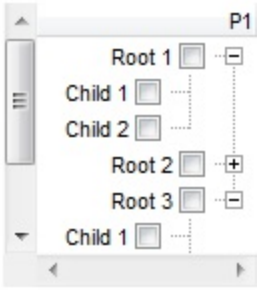
Type	Description
Boolean	A boolean expression that specifies whether the control is drawn from right to left or from left to right.

By default, the RightToLeft property is False. The RightToLeft gets or sets a value indicating whether control's elements are aligned to right or left. The RightTolLeft property affects all columns, and future columns being added.

Changing the RightToLeft property on True does the following:

- displays the vertical scroll bar on the left side of the control (Scrollbars property)
- flips the order of the columns (Position property)
- change the column's alignment to right, if the column is not centered (Alignment property, HeaderAlignment property, HeaderImageAlignment property)
- reverse the order of the drawing parts in the cells (Def(exCellDrawPartsOrder) property to "caption,picture,icons,icon,check")
- aligns the locked columns to the right (CountLockedColumns property)
- aligns the control's group-by bar / sort bar to the right (SortBarVisible property)
- the control's filter bar/prompt/close is aligned to the right (FilterBarPromptVisible property)

The following screen shot shows how the control looks if the RightToLeft property is True:



(By default) Changing the RightToLeft property on False does the following:

- displays the vertical scroll bar on the right side of the control (Scrollbars property)
- flips the order of the columns (Position property)
- change the column's alignment to left, if the column is not centered (Alignment property, HeaderAlignment property, HeaderImageAlignment property)
- reverse the order of the drawing parts in the cells (Def(exCellDrawPartsOrder) property to "check,icon,icons,picture,caption")
- aligns the locked columns to the left (CountLockedColumns property)
- aligns the control's group-by bar / sort bar to the left (SortBarVisible property)

- the control's filter bar/prompt/close is aligned to the left (FilterBarPromptVisible property)

method View.Scroll (Type as ScrollEnum, [ScrollTo as Variant])

Scrolls the control's content.

Type	Description
Type as ScrollEnum	A ScrollEnum expression that indicates type of scrolling being performed.
ScrollTo as Variant	A long expression that indicates the position where the control is scrolled when Type is exScrollVTo or exScrollHTo. If the ScrollTo parameter is missing, 0 value is used.

Use the Scroll method to scroll the control's content by code. Use the Scrollbars property specifies which scroll bars will be visible on the control. Use the ScrollPos property to get the control's scroll position. Use the EnsureVisibleItem method to ensure that a specified item fits the control's client area. Use the EnsureVisibleColumn method to ensure that a specified column fits the control's client area. If the Type parameter is exScrollLeft, exScrollRight or exScrollHTo the Scroll method scrolls horizontally the control's content pixel by pixel, if the ContinueColumnScroll property is False, else the Scroll method scrolls horizontally the control's content column by column. Use the ScrollPartVisible property to add buttons to the control's scrollbars. Use the Background property to change the visual appearance of the control's scrollbars.

property View.ScrollBars as ScrollBarsEnum

Returns or sets a value that determines whether the control has horizontal and/or vertical scroll bars.

Type	Description
ScrollBarsEnum	A ScrollBarsEnum expression that indicates which scroll bars will be visible in the control.

By default, the control adds scroll bars when required. For instance, If the ColumnAutoResize property is False and the width of the visible columns exceeds the width of the control's client area, the control shows the horizontal scroll bar. Use the ScrollBars property to hide the control's scroll bars. If the ColumnAutoResize property is True, the control does not display the control's horizontal scroll bar. Use the ScrollBySingleLine property to let users scroll the control's content item by item. Use the ContinueColumnScroll property to specify whether the user scrolls the control's content column by column or pixel by pixel. Use the EnsureVisibleItem method to ensure that an item fits the control's client area. Use the EnsureVisibleColumn method to ensure that a specified column fits the control's client area. Use the Scroll method to scroll programmatically the control. The ScrollBars property doesn't indicate whether the control displays a scroll bar. Instead, the WS_HSCROLL and WS_VSCROLL window styles indicate whether the window displays a scroll bar. Use the hWnd property to determine the handle of the control's window.

property View.ScrollBySingleLine as Boolean

Retrieves or sets a value that indicates whether the control scrolls the lines to the end. If you have at least a cell that has SingleLine false, you have to check the ScrollBySingleLine property.

Type	Description
Boolean	A boolean expression that indicates whether the control scrolls the lines one by one.

By default, the ScrollBySingleLine property is False. We recommend to set the ScrollBySingleLine property on True if you have one of the following:

- If you have at least a cell that has CellSingleLine property on false
- If your control contains at least an item that hosts an ActiveX control. See InsertControlItem property.
- If the control displays items with different height. Use the ItemHeight property to specify the item's height.

Use the EnsureVisibleItem property to ensure that an item fits the control's client area. Use the ScrollBars property to hide the control's scroll bars. Use the Scroll method to programmatically scroll the control's content.

property View.ScrollPos(Vertical as Boolean) as Long

Specifies the vertical/horizontal scroll position.

Type	Description
Vertical as Boolean	A boolean expression that specifies the scrollbar being requested. True indicates the Vertical scroll bar, False indicates the Horizontal scroll bar.
Long	A long expression that defines the scroll bar position.

Use the ScrollPos property to change programmatically the position of the control's scroll bar. Use the ScrollPos property to get the horizontal or vertical scroll position.Use the ScrollBars property to define the control's scroll bars. Use the Scroll method to scroll programmatically the control's content.

property View.SearchColumnIndex as Long

Retrieves or sets a value indicating the index of the column that is used by the auto search feature.

Type	Description
Long	A long expression that indicates the index of searching column.

Use the SearchColumnIndex property to change the searching column. The control changes the searching column when the user clicks on a column or when the user presses the TAB key (in this case the UseTabKey property should be True). If the user starts typing characters in the searching column, the control selects the item that matches the typed characters. If you want to disable the auto search feature, you have to set the SearchColumnIndex property to -1. Use the MarkSearchColumn property to hide the marker of the searching column. If the searching column is moved, the focused column is moved too.

property View.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 how the selected items are painted.

By default, the SelBackMode property is exOpaque. Use the SelBackMode property to specify how the selection appears. Use the SelBackColor property to specify the selection background color. Use the SelForeColor property to specify the selection foreground color.

property View.Select as String

Selects the path

Type	Description
String	A String expression that defines the path of selected items, using the Key column in each view.

The view's Select property selects items within the view and its descendents. The Key property indicates the column that defines the key of the view. Based on the key, and the current selection the next view is created. The Select property can select items using wild characters such as * or ?, if the view's SingleSel property is False. The Key property can be specified also through Key field of the control's DataSource property.

property View.SelectColumnIndex as Long

Retrieves or sets a value that indicates the index of the selected column, if the FullRowSelect property is False.

Type	Description
Long	A long expression that indicates the index of selected column.

The property has effect only if the FullRowSelect property is False. Use the SelectedItem property to determine the selected items. Use the SplitCell property to split a cell. Use the SelectableItem property to specify the user can select an item.

property View.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. The SelectOnRelease property has no effect if the SingleSel property is False.

property View.ShowFocusRect as Boolean

Retrieves or sets a value indicating whether the control draws a thin rectangle around the focused item.

Type	Description
Boolean	A boolean expression that indicates whether the marker for the focused cell is visible or hidden.

Use the ShowFocusRect property to hide the rectangle drawn around the focused item. Use the FocusItem property to get the focused item. If there is no focused item the FocusItem property retrieves 0. At one moment, only one item can be focused. When the selection is changed the focused item is changed too. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. If the control supports only single selection, you can use the FocusItem property to get the selected/focused item because they are always the same.

property View.ShowLockedItems as Boolean

Retrieves or sets a value that indicates whether the locked/fixed items are visible or hidden.

Type	Description
Boolean	A boolean expression that specifies whether the locked items are shown or hidden.

A locked or fixed item is always displayed on the top or bottom side of the control no matter if the control's list is scrolled up or down. Use the ShowLockedItems property to show or hide the locked items. Use the LockedItemCount property to add or remove items fixed/locked to the top or bottom side of the control. Use the LockedItem property to access a locked item by its position. Use the CellValue property assign a value to a cell.

property View.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 FocusItem to retrieve the handle of the focused item. If the control supports single selection, the FocusItem property gets the handle of the selected item too. The SelectedItem and SelectCount properties get the collection of selected items. Use the SelectItem property to programmatically select an item giving its handle. The control fires ViewItemStateStartChanging(exActivateItem) / ViewItemStateEndChanging(exActivateItem) event when the selection is changed. Use the SelBackColor and SelForeColor properties to specify the background and foreground colors for selected items. Use the SelectableItem property to specify the user can select an item. The FullRowSelect property specifies whether the selection spans the entire width of the control. the SelectAll method to select all visible items, when the control supports multiple selection. The SelectPos property selects/unselects items by position. The Selection property selects/unselects items by index.

property View.SingleSort as Boolean

Returns or sets a value that indicates whether the control supports sorting by single or multiple columns.

Type	Description
Boolean	A boolean expression that indicates whether the control supports sorting by single or multiple columns.

Use the SingleSort property to allow sorting by multiple columns. Sorting by a single column in the control is a simple matter of clicking on the column head. Sorting by multiple columns, however, is not so obvious. But it's actually quite easy. The user has two options to sort by multiple columns:

- First, sort by the first criterion, by clicking on the column head. Then hold the SHIFT key down as you click on a second heading.
- Click the column head and drag to the control's sort bar in the desired position.

By default, the SingleSort property is True, and so the user can have sorting by a single column only. Use the SortBarVisible property to show the control's sort bar. The SingleSort property is automatically set on False, if the SortBarVisible property is set to True. Use the SortOnClick property to specify the action that control should execute when the user clicks the control's header. Use the SortOrder property to sort a column programmatically. Use the SortPosition property to specify the position of the column in the sorted columns list. Use the ItemBySortPosition property to get the columns being sorted in their order.

property View.SortBarCaption as String

Specifies the caption being displayed on the control's sort bar when the sort bar contains no columns.

Type	Description
String	A String expression that indicates the caption of the control's sort bar.

The SortBarCaption property specifies the caption of the control's sort bar, when it contains no sorted columns. Use the SortBarVisible property to show the control's sort bar. Use the BackColorSortBar, BackColorSortBarCaption and ForeColorSortBar properties to specify colors for the control's sort bar. Use the SortBarHeight property to specify the height of the control's sort bar. Use the SortBarColumnWidth property to specify the width of the column in the control's sort bar. By default, the SortBarCaption property is "Drag a column header here to sort by that column.". Use the Font property to specify the control's font. Use the ItemBySortPosition property to access the columns in the control's sort bar.

The SortBarCaption property may include built-in HTML tags 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 $_{subscript}$ 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:

- **<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 View.SortBarColumnWidth as Long

Specifies the maximum width a column can be in the control's sort bar.

Type	Description
Long	A long expression that indicates the width of the columns in the control's sort bar. If the value is negative, all columns in the sort bar are displayed with the same width (the absolute value represents the width of the columns, in pixels). If the value is positive, it indicates the maximum width, so the width of the columns in the sort bar may differ.

the SortBarColumnWidth property to specify the width of the column in the control's sort bar. Use the SortBarVisible property to show the control's sort bar. Use the Width property to specify the width of the column in the control's header bar. Use the SortBarHeight property to specify the height of the control's sort bar. Use the SortBarCaption property to specify the caption being displayed in the control's sort bar when it contains no columns.

property View.SortBarHeight as Long

Retrieves or sets a value that indicates the height of the control's sort bar.

Type	Description
Long	A long expression that indicates the height of the control's sort bar, in pixels.

Use the SortBarHeight property to specify the height of the control's sort bar. Use the SortBarVisible property to show the control's sort bar. By default, the SortBarHeight property is 18 pixels. Use the HeaderHeight property to specify the height of the control's header bar. Use the SortBarColumnWidth property to specify the width of the columns being displayed in the control's sort bar. Use the BackColorSortBar, BackColorSortBarCaption and ForeColorSortBar properties to specify colors for the control's sort bar. Use the SortBarCaption property to specify the caption being displayed in the control's sort bar when it contains no columns.

property View.SortBarVisible as Boolean

Retrieves or sets a value that indicates whether control's sort bar is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the sort bar is visible or hidden.

Use the SortBarVisible property to show the control's sort bar. By default, the SortBarVisible property is False. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. Sorting by a single column in the control is a simple matter of clicking on the column head. Sorting by multiple columns, however, is not so obvious. But it's actually quite easy. The user has two options to sort by multiple columns:

- First, sort by the first criterion, by clicking on the column head. Then hold the SHIFT key down as you click on a second heading.
- Click the column head and drag to the control's sort bar in the desired position.

property View.SortOnClick as SortOnClickEnum

Retrieves or sets a value that indicates whether the control automatically sorts the data when the user clicks on a column's caption.

Type	Description
SortOnClickEnum	A SortOnClickEnum expression that indicates the action that control takes whether the user clicks the column's header.

Use the SortOnClick property to disable sorting items when the user clicks on the column's header. Use the SortBarVisible property to show the control's sort bar. Use the SingleSort property to allow sorting by single or multiple columns. Use the AllowSort property to avoid sorting a column when user clicks the column. Use the DefaultSortOrder property to specify the column's default sort order, when the user first clicks the column's header. Use the SortChildren method to sort a column, at runtime. Use the DisplaySortIcon property to hide the sort icon if the column is sorted. Use the HeaderVisible property to show or hide the control's header. Use the BackColorHeader property to specify the header's background color. Use the AllowSizing property to disable resizing a column when user clicks the right margin of the column.

There are two methods to get the items sorted like follows:

- Using the SortOrder property of the Column object::

  ```
  View1.Columns(ColIndex).SortOrder = SortAscending
  ```

  The SortOrder property adds the sorting icon to the column's header, if the DisplaySortIcon property is True.

- Using the SortChildren method of the Items collection. The SortChildren sorts the items. The SortChildren method sorts the child items of the given parent item in the control. SortChildren will not recourse through the tree, only the immediate children of the item will be sorted. The following sample sorts descending the list of root items on the "Column 1"(if your control displays a list, all items are considered being root items).

  ```
  View1.Items.SortChildren 0, "Column 1", False
  ```

property View.Tag as Variant

Specifies any extra data associated with the view.

Type	Description
Variant	A Variant expression associated with the view.

The Tag property associates any extra data to the current view. The View property gets the view giving its index or tag. The DataSource property can change the Tag property using the Tag field as explained bellow:

- **Tag** or **Data Tag**, specifies any extra data associated with the view. For instance, "**Tag**=Country". The Tag property of the View can be used to access the tag of the view at runtime. The Tag field is not required.

The Select property can select items using wild characters such as * or ?, if the view's SingleSel property is False.

property View.TreeColumnIndex as Long

Retrieves or sets a value that indicates the index of column where the hierarchy lines are displayed.

Type	Description
Long	A long expression that indicates the index of column that displays the control's hierarchy.

Use the TreeColumnIndex property to change the column's index where the hierarchy lines are painted. Use HasLines and LinesAtRoot properties to show the hierarchy lines. Use the HasButtons property to define the +/- signs appearance. If the TreeColumnIndex property is -1, the control doesn't paint the hierarchy. Use the Indent property to define the amount, in pixels, that child items are indented relative to their parent items. Use the InsertItem property to insert child items.

method View.Ungroup ()

Ungroups the columns, if they have been previously grouped.

Type	Description

The Ungroup method removes the grouping items from the control's list. The AllowGroupBy property specifies whether the control supports Group-By feature. The Ungroup method has no effect if the AllowGroupBy property is False, or no columns is grouped. The Group method forces the control to re-group the items. During execution any of these methods, the IsGrouping property returns True. You can call the SortOrder property to sort and group by specified column. Use the SortType property to determine the way how the column is sorted. The *ViewItemUpdate(exAddGroupItem)* event is fired when a new grouping items is added to the control's list. *You can use the AddGroupItem event, to add headers or footers during grouping, customize the aggregate formula to be displayed on different columns, while dropping a column to the sortbar.* The Column.AllowGroupBy property may be used to prevent grouping a specific column. The AllowSort property indicates whether the user can sort a column by clicking the column's header.

property View.Value ([Column as Variant]) as Variant

Indicates the value of the single active item on the specified column.

Type	Description
Column as Variant	A long expression / string expression that specifies the column where the value is being requested.
Variant	A VARIANT expression that specifies the selected value in giving column. The CellValue property specifies the cell's value.

The Value property returns the value of the single active item on the specified column. The Values property returns a safe array with all values of selected / active items in the view, on the specified column. The ValueList property returns the list of values for all selected / active items in the view, on the specified column, separated by comma.

As Microsoft Access uses DAO, you need to use the View's DataSource property rather than control's DataSource property as in the following sample:

```
Private Sub CascadeTree1_CreateView(ByVal View As Object)
   With View
      Select Case .Index
         Case 1: ' State or City
            .DataSource = CurrentDb.OpenRecordset("Select * FROM States WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
            .Tag = "State"
            .Key = "StateCode"
            .Name = "StateName"
            If (.Items.ItemCount = 0) Then
               .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
               .Tag = "City"
               .Key = ""
               .Name = "Name"
               .ColumnAutoResize = False
            End If
         Case 2: ' City
            .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ParentView.ValueList("CountryCode") & ") AND
StateCode IN (" & .ParentView.ValueList("StateCode") & ")")
```

```
            .Tag = "City"
            .Key = ""
            .Name = "Name"
        End Select
    End With
End Sub

Private Sub Form_Load()
    With CascadeTree1.DefaultView
    .DataSource = CurrentDb.OpenRecordset("SELECT * FROM Countries")
    .Tag = "Country"
    .Key = "CountryCode"
    .Name = "CountryName"
    End With
End Sub
```

The sample loads the Countries table into the default view (view with the index 0). Once the user clicks / selects / activates an item, the control creates a new view (with the index 1, 2 and so on) and fires the CreateView event. During the CreateView event you can load data from different tables based on the parent's view selection. See the ParentView.ValueList

property View.ValueList ([Column as Variant]) as String

Returns the list of values for all selected / active items in the view, on the specified column, separated by comma.

Type	Description
Column as Variant	A long expression / string expression that specifies the column where the value is being requested.
String	A String expression that specifies the selected values, separated by , (comma) character. Each string value is returned between " characters, while a date between ## characters,

The ValueList property returns the list of values for all selected / active items in the view, on the specified column, separated by comma. The Value property returns the value of the single active item on the specified column. The Values property returns a safe array with all values of selected / active items in the view, on the specified column.

As Microsoft Access uses DAO, you need to use the View's DataSource property rather than control's DataSource property as in the following sample:

```
Private Sub CascadeTree1_CreateView(ByVal View As Object)
   With View
      Select Case .Index
         Case 1: ' State or City
            .DataSource = CurrentDb.OpenRecordset("Select * FROM States WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
            .Tag = "State"
            .Key = "StateCode"
            .Name = "StateName"
            If (.Items.ItemCount = 0) Then
               .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
               .Tag = "City"
               .Key = ""
               .Name = "Name"
               .ColumnAutoResize = False
            End If
         Case 2: ' City
            .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
```

```
CountryCode IN (" & .ParentView.ParentView.ValueList("CountryCode") & ") AND
StateCode IN (" & .ParentView.ValueList("StateCode") & ")")
            .Tag = "City"
            .Key = ""
            .Name = "Name"
        End Select
    End With
End Sub

Private Sub Form_Load()
    With CascadeTree1.DefaultView
    .DataSource = CurrentDb.OpenRecordset("SELECT * FROM Countries")
    .Tag = "Country"
    .Key = "CountryCode"
    .Name = "CountryName"
    End With
End Sub
```

The sample loads the Countries table into the default view (view with the index 0). Once the user clicks / selects / activates an item, the control creates a new view (with the index 1, 2 and so on) and fires the CreateView event. During the CreateView event you can load data from different tables based on the parent's view selection. See the ParentView.ValueList

property View.Values ([Column as Variant]) as Variant

Returns a safe array with all values of selected / active items in the view, on the specified column.

Type	Description
Column as Variant	A long expression / string expression that specifies the column where the value is being requested.
Variant	A safe array with all values of selected / active items in the view, on the specified column.

The Values property returns a safe array with all values of selected / active items in the view, on the specified column. The ValueList property returns the list of values for all selected / active items in the view, on the specified column, separated by comma. The Value property returns the value of the single active item on the specified column.

As Microsoft Access uses DAO, you need to use the View's DataSource property rather than control's DataSource property as in the following sample:

```
Private Sub CascadeTree1_CreateView(ByVal View As Object)
   With View
      Select Case .Index
         Case 1: ' State or City
            .DataSource = CurrentDb.OpenRecordset("Select * FROM States WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
            .Tag = "State"
            .Key = "StateCode"
            .Name = "StateName"
            If (.Items.ItemCount = 0) Then
               .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ValueList("CountryCode") & " )")
               .Tag = "City"
               .Key = ""
               .Name = "Name"
               .ColumnAutoResize = False
            End If
         Case 2: ' City
            .DataSource = CurrentDb.OpenRecordset("Select * FROM Cities WHERE
CountryCode IN (" & .ParentView.ParentView.ValueList("CountryCode") & ") AND
StateCode IN (" & .ParentView.ValueList("StateCode") & ")")
```

```
            .Tag = "City"
            .Key = ""
            .Name = "Name"
        End Select
    End With
End Sub

Private Sub Form_Load()
    With CascadeTree1.DefaultView
    .DataSource = CurrentDb.OpenRecordset("SELECT * FROM Countries")
    .Tag = "Country"
    .Key = "CountryCode"
    .Name = "CountryName"
    End With
End Sub
```

The sample loads the Countries table into the default view (view with the index 0). Once the user clicks / selects / activates an item, the control creates a new view (with the index 1, 2 and so on) and fires the CreateView event. During the CreateView event you can load data from different tables based on the parent's view selection. See the ParentView.ValueList

property View.View ([Tag as Variant]) as View

Gets the view giving its index or tag.

Type	Description
Tag as Variant	A VARIANT expression that specifies the Tag of the view being searched
View	A View object by tag.

The View property gets the view giving its index or tag. The Tag property associates any extra data to the current view. The DataSource property can change the Tag property using the Tag field as explained bellow:

- **Tag** or **Data Tag**, specifies any extra data associated with the view. For instance, "**Tag**=Country". The Tag property of the View can be used to access the tag of the view at runtime. The Tag field is not required.

The Select property can select items using wild characters such as * or ?, if the view's SingleSel property is False.

property View.Width as Long

Specifies the width of the view.

Type	Description
Long	A Long expression that specifies the width of the view.

The Width property specifies the width of the view. The WidthToFit property specifies the width of the view to fit the control's client area. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property View.WidthToFit as Long

Specifies the width of the view to fit the control's client area.

Type	Description
Long	A long expression that specifies the width of the view to fit the control's client area.

The WidthToFit property specifies the width of the view to fit the control's client area. The Width property specifies the width of the view. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

ExCascadeTree events

The CascadeTree object 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.
CreateView	A view has been created.
DblClick	Occurs when the user dblclk the left mouse button over an object.
DestroyView	A view requires to be destroyed.
Error	An internal error occurs.
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.
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.
ViewEndChanging	Occurs once the user is about to change the view.
ViewItemStateEndChanging	Indicates that the state of the item has been changed.
ViewItemStateStartChanging	Indicates that the state of the item is about to be changed.
ViewItemUpdate	Indicates that an item has been added or removed from the working view.
ViewStartChanging	Occurs once the user is about to change the view.

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. The View property specifies the view where the event occurs.

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,
AxEXCASCADETREELib._IGaugeEvents_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:
AxEXCASCADETREELib._IGaugeEvents_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
AxEXCASCADETREELib._IGaugeEvents_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(oGauge,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 clicks the list.

Type	Description

Use the Click event to notify your application when the user clicks the list. Use the MouseDown or MouseUp event to notify your application when the user presses or releases the one of the mouse buttons. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

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(oCascadeTree)
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 CreateView (View as View)

A view has been created.

Type	Description
View as View	A View object being created. The view parameter is equivalent with the View property.

The CreateView event is fired as soon as a new item has been selected in a previously view. The CreateView event can be used to initialize the view once the user activates an item. The ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) events notify your application that an item has been selected.

The following properties can be used to access a view:

- FirstView property, gets the first view
- PrevView property, gets the previously view (parent)
- ParentView property, gets the parent view (previously)
- ChildView property, gets the child view (next).
- NextView property, gets the next view (child).

- LastView property, gets the last view.

Syntax for CreateView event, **/NET** version, on:

C#
```
private void CreateView(object sender,exontrol.EXCASCADETREELib.View   View)
{
}
```

VB
```
Private Sub CreateView(ByVal sender As System.Object,ByVal View As
exontrol.EXCASCADETREELib.View) Handles CreateView
End Sub
```

Syntax for CreateView event, **/COM** version, on:

C#
```
private void CreateView(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_CreateViewEvent e)
{
}
```

C++
```
void OnCreateView(LPDISPATCH   View)
{
```

```
}
```

C++ Builder

```
void __fastcall CreateView(TObject *Sender,Excascadetreelib_tlb::IView   *View)
{
}
```

Delphi

```
procedure CreateView(ASender: TObject; View : IView);
begin
end;
```

Delphi 8 (.NET only)

```
procedure CreateView(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_CreateViewEvent);
begin
end;
```

Powe…

```
begin event CreateView(oleobject  View)

end event CreateView
```

VB.NET

```
Private Sub CreateView(ByVal sender As System.Object, ByVal e As
AxEXCASCADETREELib._ICascadeTreeEvents_CreateViewEvent) Handles CreateView
End Sub
```

VB6

```
Private Sub CreateView(ByVal View As EXCASCADETREELibCtl.IView)
End Sub
```

VBA

```
Private Sub CreateView(ByVal View As Object)
End Sub
```

VFP

```
LPARAMETERS View
```

Xbas…

```
PROCEDURE OnCreateView(oCascadeTree,View)

RETURN
```

Syntax for CreateView event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="CreateView(View)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function CreateView(View)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComCreateView Variant   llView
    Forward Send OnComCreateView llView
End_Procedure
```

Visual Objects

```
METHOD OCX_CreateView(View) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_CreateView(COM   _View)
{
}
```

XBasic

```
function CreateView as v (View  as  OLE::Exontrol.CascadeTree.1::IView)
end function
```

dBASE

```
function nativeObject_CreateView(View)
return
```

event DblClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Fired when the user double clicks an item.

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 DblClk event is fired whenever the user double clicks a file or a folder. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

Syntax for DblClick event, **/NET** version, on:

C#
```
private void DblClick(object sender)
{
}
```

VB
```
Private Sub DblClick(ByVal sender As System.Object) Handles DblClick
End Sub
```

Syntax for DblClick event, **/COM** version, on:

C#
```
private void DblClick(object sender, EventArgs e)
{
}
```

C++
```
void OnDblClick()
{
}
```

C++ Builder

```
void __fastcall DblClick(TObject *Sender)
{
}
```

Delphi

```
procedure DblClick(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure DblClick(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event DblClick()
end event DblClick
```

VB.NET

```
Private Sub DblClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DblClick
End Sub
```

VB6

```
Private Sub DblClick()
End Sub
```

VBA

```
Private Sub DblClick()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnDblClick(oCascadeTree)
RETURN
```

Syntax for DblClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="DblClick()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DblClick()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDblClick
    Forward Send OnComDblClick
End_Procedure
```

Visual Objects

```
METHOD OCX_DblClick() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DblClick()
{
}
```

XBasic

```
function DblClick as v ()
end function
```

dBASE

```
function nativeObject_DblClick()
return
```

event DestroyView (View as View)

A view requires to be destroyed.

Type	Description
View as View	A View object being destroyed. The view parameter is equivalent with the View property.

The DestroyView event can be used to release any extra data associated with the view. The ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) events notify your application that an item has been selected.

Syntax for DestroyView event, **/NET** version, on:

C#
```
private void DestroyView(object sender,exontrol.EXCASCADETREELib.View   View)
{
}
```

VB
```
Private Sub DestroyView(ByVal sender As System.Object,ByVal View As
exontrol.EXCASCADETREELib.View) Handles DestroyView
End Sub
```

Syntax for DestroyView event, **/COM** version, on:

C#
```
private void DestroyView(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_DestroyViewEvent e)
{
}
```

C++
```
void OnDestroyView(LPDISPATCH   View)
{
}
```

C++ Builder
```
void __fastcall DestroyView(TObject *Sender,Excascadetreelib_tlb::IView   *View)
{
}
```

Delphi
```
procedure DestroyView(ASender: TObject; View : IView);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DestroyView(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_DestroyViewEvent);
begin
end;
```

Powe…

```
begin event DestroyView(oleobject  View)

end event DestroyView
```

VB.NET

```
Private Sub DestroyView(ByVal sender As System.Object, ByVal e As
AxEXCASCADETREELib._ICascadeTreeEvents_DestroyViewEvent) Handles
DestroyView
End Sub
```

VB6

```
Private Sub DestroyView(ByVal View As EXCASCADETREELibCtl.IView)
End Sub
```

VBA

```
Private Sub DestroyView(ByVal View As Object)
End Sub
```

VFP

```
LPARAMETERS View
```

Xbas…

```
PROCEDURE OnDestroyView(oCascadeTree,View)

RETURN
```

Syntax for DestroyView event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="DestroyView(View)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DestroyView(View)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDestroyView Variant   llView
   Forward Send OnComDestroyView llView
End_Procedure
```

Visual Objects

```
METHOD OCX_DestroyView(View) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DestroyView(COM   _View)
{
}
```

XBasic

```
function DestroyView as v (View  as  OLE::Exontrol.CascadeTree.1::IView)
end function
```

dBASE

```
function nativeObject_DestroyView(View)
return
```

event Error (Error as Long, Description as String)

An internal error occurs.

Type	Description
Error as Long	A long expression that specifies the code of the error
Description as String	A String expression that specifies the description of the error.

The Error event notifies your application once an error occurs. The DataSource property specifies the control's data as an array, XML, ADO or DAO. The View property specifies the view where the event occurs.

Syntax for Error event, **/NET** version, on:

C#

```
private void Error(object sender,int   Err,string   Description)
{
}
```

VB

```
Private Sub Error(ByVal sender As System.Object,ByVal Err As Integer,ByVal
Description As String) Handles Error
End Sub
```

Syntax for Error event, **/COM** version, on:

C#

```
private void Error(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_ErrorEvent e)
{
}
```

C++

```
void OnError(long   Error,LPCTSTR   Description)
{
}
```

C++ Builder

```
void __fastcall Error(TObject *Sender,long   Error,BSTR   Description)
{
}
```

Delphi

```
procedure Error(ASender: TObject; Error : Integer;Description : WideString);
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure Error(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_ErrorEvent);
begin
end;
```

Powe…

```
begin event Error(long  Error,string  Description)


end event Error
```

VB.NET

```
Private Sub Error(ByVal sender As System.Object, ByVal e As
AxEXCASCADETREELib._ICascadeTreeEvents_ErrorEvent) Handles Error
End Sub
```

VB6

```
Private Sub Error(ByVal Error As Long,ByVal Description As String)
End Sub
```

VBA

```
Private Sub Error(ByVal Error As Long,ByVal Description As String)
End Sub
```

VFP

```
LPARAMETERS Error,Description


```

Xbas…

```
PROCEDURE OnError(oCascadeTree,Error,Description)

RETURN
```

Syntax for Error event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Error(Error,Description)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Error(Error,Description)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComError Integer   llError String   llDescription
    Forward Send OnComError llError llDescription
End_Procedure
```

Visual Objects

```
METHOD OCX_Error(Error,Description) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Error(int   _Error,str   _Description)
{
}
```

XBasic

```
function Error as v (Error  as  N,Description  as  C)
end function
```

dBASE

```
function nativeObject_Error(Error,Description)
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. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties).

The Event notification occurs ANY time the control fires an event. The View property specifies the view where the event occurs.

This is useful for X++ language, which does not support event with parameters passed by reference.

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.

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,
AxEXCASCADETREELib._ICascadeTreeEvents_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:
AxEXCASCADETREELib._ICascadeTreeEvents_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
AxEXCASCADETREELib._ICascadeTreeEvents_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(oCascadeTree,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
```

The solution is using and handling the Event notification and EventParam method., instead handling the event that gives the "invalid parameters" error executing code.

Let's presume that we need to handle the BarParentChange event to change the _Cancel parameter from false to true, which fires the "Error executing code: FormActiveXControl (data source), method onEvent_BarParentChange called with invalid parameters." We need to know the identifier of the BarParentChange event (each event has an unique identifier and it is static, defined in the control's type library). 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 cascadetree1.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
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" 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.

Now, we can see that the identifier for the BarParentChange event is 125, so we need to handle the Event event as:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
        cascadetree1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

The code checks if the BarParentChange (_EventID == 125) event is fired, and changes the third parameter of the event to true. The definition for BarParentChange event can be consulted in the control's documentation or in the ActiveX explorer. So, anytime you need to access the original parameters for the event you should use the EventParam method that allows you to get or set a parameter. If the parameter is not passed by reference, you can not change the parameter's value.

Now, let's add some code to see a complex sample, so let's say that we need to prevent moving the bar from an item to any disabled item. So, we need to specify the Cancel parameter as not Items.EnableItem(NewItem), in other words cancels if the new parent is disabled. Shortly the code will be:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem
```

```
as HITEM, Cancel as Boolean) */
        if ( !cascadetree1.Items().EnableItem( cascadetree1.EventParam( 2 /*NewItem*/ ) ) )
            cascadetree1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

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

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. The View property specifies the view where the event occurs. 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,
AxEXCASCADETREELib._ICascadeTreeEvents_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:
AxEXCASCADETREELib._ICascadeTreeEvents_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
AxEXCASCADETREELib._ICascadeTreeEvents_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(oCascadeTree,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. The View property specifies the view where the event occurs.

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,
AxEXCASCADETREELib._ICascadeTreeEvents_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:
AxEXCASCADETREELib._ICascadeTreeEvents_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
AxEXCASCADETREELib._ICascadeTreeEvents_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(oCascadeTree,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)

Occur 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. The View property specifies the view where the event occurs.

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,
AxEXCASCADETREELib._ICascadeTreeEvents_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:
AxEXCASCADETREELib._ICascadeTreeEvents_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
AxEXCASCADETREELib._ICascadeTreeEvents_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(oCascadeTree,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 MouseDown (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occur 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 the 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 View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

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,
AxEXCASCADETREELib._ICascadeTreeEvents_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:
AxEXCASCADETREELib._ICascadeTreeEvents_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
AxEXCASCADETREELib._ICascadeTreeEvents_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(oCascadeTree,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.CascadeTree.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.CascadeTree.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 View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column 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,
AxEXCASCADETREELib._ICascadeTreeEvents_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:
AxEXCASCADETREELib._ICascadeTreeEvents_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
AxEXCASCADETREELib._ICascadeTreeEvents_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(oCascadeTree,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.CascadeTree.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.CascadeTree.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. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

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,
AxEXCASCADETREELib._ICascadeTreeEvents_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:
AxEXCASCADETREELib._ICascadeTreeEvents_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
AxEXCASCADETREELib._ICascadeTreeEvents_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(oCascadeTree,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.CascadeTree.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.CascadeTree.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseUp(Button,Shift,X,Y)
return
```

event RClick ()

Occurs once the user right clicks the control.

Type	Description

Notifies your application once the user right-clicks the control. Use the MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

event ViewEndChanging (Operation as ViewOperationEnum)

Occurs once the user is about to change the view.

Type	Description
Operation as ViewOperationEnum	A ViewOperationEnum expression that specifies the operation that ended.

The ViewStartChanging / ViewEndChanging events notify your application that an operation starts or ends. For instance, ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) events notify your application that an item has been selected. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

Syntax for ViewEndChanging event, **/NET** version, on:

C#

```
private void ViewEndChanging(object
sender,exontrol.EXCASCADETREELib.ViewOperationEnum   Operation)
{
}
```

VB

```
Private Sub ViewEndChanging(ByVal sender As System.Object,ByVal Operation As
exontrol.EXCASCADETREELib.ViewOperationEnum) Handles ViewEndChanging
End Sub
```

Syntax for ViewEndChanging event, **/COM** version, on:

C#

```
private void ViewEndChanging(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_ViewEndChangingEvent e)
{
}
```

C++

```
void OnViewEndChanging(long   Operation)
{
}
```

C++ Builder

```
void __fastcall ViewEndChanging(TObject
*Sender,Exmillerlib_tlb::ViewOperationEnum   Operation)
{
```

```
}
```

Delphi

```
procedure ViewEndChanging(ASender: TObject; Operation : ViewOperationEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ViewEndChanging(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_ViewEndChangingEvent);
begin
end;
```

Powe…

```
begin event ViewEndChanging(long  Operation)

end event ViewEndChanging
```

VB.NET

```
Private Sub ViewEndChanging(ByVal sender As System.Object, ByVal e As
AxEXCASCADETREELib._ICascadeTreeEvents_ViewEndChangingEvent) Handles
ViewEndChanging
End Sub
```

VB6

```
Private Sub ViewEndChanging(ByVal Operation As
EXCASCADETREELibCtl.ViewOperationEnum)
End Sub
```

VBA

```
Private Sub ViewEndChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation

```

Xbas…

```
PROCEDURE OnViewEndChanging(oCascadeTree,Operation)

RETURN
```

Syntax for ViewEndChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ViewEndChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ViewEndChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComViewEndChanging OLEViewOperationEnum   llOperation
    Forward Send OnComViewEndChanging llOperation
End_Procedure
```

Visual Objects

```
METHOD OCX_ViewEndChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ViewEndChanging(int   _Operation)
{
}
```

XBasic

```
function ViewEndChanging as v (Operation  as
OLE::Exontrol.ExMiller.1::ViewOperationEnum)
end function
```

dBASE

```
function nativeObject_ViewEndChanging(Operation)
return
```

event ViewItemStateEndChanging (Operation as ViewItemStateEnum, Item as HITEM, ColIndex as Long)

Indicates that the state of the item has been changed.

Type	Description
Operation as ViewItemStateEnum	A ViewItemStateEnum expression that specifies the item operation that ends.
Item as HITEM	A Long expression that specifies the handle of the item where the operation occurs. The View property specifies the view where the event occurs. The Items property of the View object gives access to the items collection of the view.
ColIndex as Long	A Long expression that specifies the index of the column, where the operation occurs. The View property specifies the view where the event occurs. The Colums property of the View object gives access to the view's Columns collection. For instance, if the cell's check-box state is changing the ColIndex parameter specifies index of the column where check-box has been clicked.

The ViewItemStateStartChanging / ViewItemStateEndChanging notifies your application that an item expanded or activated / selected, or when a check box has been clicked / changed. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

Syntax for ViewItemStateEndChanging event, **/NET** version, on:

C#
```
private void ViewItemStateEndChanging(object
sender,exontrol.EXCASCADETREELib.ViewItemStateEnum  Operation,int  Item,int
ColIndex)
{
}
```

VB
```
Private Sub ViewItemStateEndChanging(ByVal sender As System.Object,ByVal
Operation As exontrol.EXCASCADETREELib.ViewItemStateEnum,ByVal Item As
Integer,ByVal ColIndex As Integer) Handles ViewItemStateEndChanging
End Sub
```

Syntax for ViewItemStateEndChanging event, **/COM** version, on:

C#
```
private void ViewItemStateEndChanging(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemStateEndChangingEvent e)
{
}
```

C++
```
void OnViewItemStateEndChanging(long  Operation,long  Item,long  ColIndex)
{
}
```

C++ Builder
```
void __fastcall ViewItemStateEndChanging(TObject
*Sender,Excascadetreelib_tlb::ViewItemStateEnum
Operation,Excascadetreelib_tlb::HITEM  Item,long  ColIndex)
{
}
```

Delphi
```
procedure ViewItemStateEndChanging(ASender: TObject; Operation :
ViewItemStateEnum;Item : HITEM;ColIndex : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure ViewItemStateEndChanging(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemStateEndChangingEvent);
begin
end;
```

Powe…
```
begin event ViewItemStateEndChanging(long  Operation,long  Item,long
ColIndex)


end event ViewItemStateEndChanging
```

VB.NET
```
Private Sub ViewItemStateEndChanging(ByVal sender As System.Object, ByVal e As
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemStateEndChangingEvent)
Handles ViewItemStateEndChanging
End Sub
```

VB6
```
Private Sub ViewItemStateEndChanging(ByVal Operation As
```

```
EXCASCADETREELibCtl.ViewItemStateEnum,ByVal Item As
EXCASCADETREELibCtl.HITEM,ByVal ColIndex As Long)
End Sub
```

VBA

```
Private Sub ViewItemStateEndChanging(ByVal Operation As Long,ByVal Item As
Long,ByVal ColIndex As Long)
End Sub
```

VFP

```
LPARAMETERS Operation,Item,ColIndex

```

Xbas…

```
PROCEDURE
OnViewItemStateEndChanging(oCascadeTree,Operation,Item,ColIndex)

RETURN
```

Syntax for ViewItemStateEndChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ViewItemStateEndChanging(Operation,Item,ColIndex)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ViewItemStateEndChanging(Operation,Item,ColIndex)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComViewItemStateEndChanging OLEViewItemStateEnum
llOperation HITEM   llItem Integer   llColIndex
    Forward Send OnComViewItemStateEndChanging llOperation llItem llColIndex
End_Procedure
```

Visual Objects

```
METHOD OCX_ViewItemStateEndChanging(Operation,Item,ColIndex) CLASS
MainDialog
RETURN NIL
```

X++

```
void onEvent_ViewItemStateEndChanging(int   _Operation,int   _Item,int
_ColIndex)
```

```
{
}
```

XBasic

```
function ViewItemStateEndChanging as v (Operation  as
OLE::Exontrol.CascadeTree.1::ViewItemStateEnum,Item  as
OLE::Exontrol.CascadeTree.1::HITEM,ColIndex  as  N)
end function
```

dBASE

```
function nativeObject_ViewItemStateEndChanging(Operation,Item,ColIndex)
return
```

event ViewItemStateStartChanging (Operation as ViewItemStateEnum, Item as HITEM, ColIndex as Long, Cancel as Variant)

Indicates that the state of the item is about to be changed.

Type	Description
Operation as ViewItemStateEnum	A ViewItemStateEnum expression that specifies the item operation that starts.
Item as HITEM	A Long expression that specifies the handle of the item where the operation occurs. The View property specifies the view where the event occurs. The Items property of the View object gives access to the items collection of the view.
ColIndex as Long	A Long expression that specifies the index of the column, where the operation occurs. The View property specifies the view where the event occurs. The Colums property of the View object gives access to the view's Columns collection. For instance, if the cell's check-box state is changing the ColIndex parameter specifies index of the column where check-box has been clicked.
Cancel as Variant	A Boolean expression that specifies whether the operation should be canceled.

The ViewItemStateStartChanging / ViewItemStateEndChanging notifies your application that an item expanded or activated / selected, or when a check box has been clicked / changed. The The ViewItemStateStartChanging event can be used to cancel any of the specified operations. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

Syntax for ViewItemStateStartChanging event, **/NET** version, on:

C#
```
private void ViewItemStateStartChanging(object
sender,exontrol.EXCASCADETREELib.ViewItemStateEnum  Operation,int  Item,int
ColIndex,ref object  Cancel)
{
}
```

VB
```
Private Sub ViewItemStateStartChanging(ByVal sender As System.Object,ByVal
Operation As exontrol.EXCASCADETREELib.ViewItemStateEnum,ByVal Item As
```

```
Integer,ByVal ColIndex As Integer,ByRef Cancel As Object) Handles
ViewItemStateStartChanging
End Sub
```

Syntax for ViewItemStateStartChanging event, **/COM** version, on:

C#

```
private void ViewItemStateStartChanging(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemStateStartChangingEvent e)
{
}
```

C++

```
void OnViewItemStateStartChanging(long  Operation,long  Item,long
ColIndex,VARIANT FAR*  Cancel)
{
}
```

C++ Builder

```
void __fastcall ViewItemStateStartChanging(TObject
*Sender,Excascadetreelib_tlb::ViewItemStateEnum
Operation,Excascadetreelib_tlb::HITEM  Item,long  ColIndex,Variant *  Cancel)
{
}
```

Delphi

```
procedure ViewItemStateStartChanging(ASender: TObject; Operation :
ViewItemStateEnum;Item : HITEM;ColIndex : Integer;var Cancel : OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ViewItemStateStartChanging(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemStateStartChangingEvent);
begin
end;
```

Powe…

```
begin event ViewItemStateStartChanging(long  Operation,long  Item,long
ColIndex,any  Cancel)

end event ViewItemStateStartChanging
```

VB.NET

```
Private Sub ViewItemStateStartChanging(ByVal sender As System.Object, ByVal e
```

```
As
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemStateStartChangingEvent)
Handles ViewItemStateStartChanging
End Sub
```

VB6

```
Private Sub ViewItemStateStartChanging(ByVal Operation As
EXCASCADETREELibCtl.ViewItemStateEnum,ByVal Item As
EXCASCADETREELibCtl.HITEM,ByVal ColIndex As Long,Cancel As Variant)
End Sub
```

VBA

```
Private Sub ViewItemStateStartChanging(ByVal Operation As Long,ByVal Item As
Long,ByVal ColIndex As Long,Cancel As Variant)
End Sub
```

VFP

```
LPARAMETERS Operation,Item,ColIndex,Cancel
```

Xbas…

```
PROCEDURE
OnViewItemStateStartChanging(oCascadeTree,Operation,Item,ColIndex,Cancel)

RETURN
```

Syntax for ViewItemStateStartChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ViewItemStateStartChanging(Operation,Item,ColIndex,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ViewItemStateStartChanging(Operation,Item,ColIndex,Cancel)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComViewItemStateStartChanging OLEViewItemStateEnum
llOperation HITEM   llItem Integer   llColIndex Variant   llCancel
    Forward Send OnComViewItemStateStartChanging llOperation llItem llColIndex
llCancel
End_Procedure
```

Visual Objects

```
METHOD OCX_ViewItemStateStartChanging(Operation,Item,ColIndex,Cancel)
CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ViewItemStateStartChanging(int   _Operation,int   _Item,int
_ColIndex,COMVariant /*variant*/   _Cancel)
{
}
```

XBasic

```
function ViewItemStateStartChanging as v (Operation  as
OLE::Exontrol.CascadeTree.1::ViewItemStateEnum,Item  as
OLE::Exontrol.CascadeTree.1::HITEM,ColIndex  as  N,Cancel  as  A)
end function
```

dBASE

```
function
nativeObject_ViewItemStateStartChanging(Operation,Item,ColIndex,Cancel)
return
```

event ViewItemUpdate (Operation as ViewItemUpdateEnum, Item as HITEM)

Indicates that an item has been added or removed from the working view.

Type	Description
Operation as ViewItemUpdateEnum	A ViewItemUpdateEnum that specifies the operation that occurred.
Item as HITEM	A Long expression that specifies the handle of the item that has been added or removed. The View property specifies the view where the event occurs. The Items property of the View object gives access to the items collection of the view. The Colums property of the View object gives access to the view's Columns collection.

The ViewItemUpdate event notifies your application that a new item has been added or removed of the View object. The ViewItemStateStartChanging / ViewItemStateEndChanging notifies your application that an item expanded or activated / selected, or when a check box has been clicked / changed. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

Syntax for ViewItemUpdate event, **/NET** version, on:

C#
```
private void ViewItemUpdate(object
sender,exontrol.EXCASCADETREELib.ViewItemUpdateEnum  Operation,int  Item)
{
}
```

VB
```
Private Sub ViewItemUpdate(ByVal sender As System.Object,ByVal Operation As
exontrol.EXCASCADETREELib.ViewItemUpdateEnum,ByVal Item As Integer)
Handles ViewItemUpdate
End Sub
```

Syntax for ViewItemUpdate event, **/COM** version, on:

C#
```
private void ViewItemUpdate(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemUpdateEvent e)
{
```

```
}
```

C++

```
void OnViewItemUpdate(long   Operation,long   Item)
{
}
```

C++ Builder

```
void __fastcall ViewItemUpdate(TObject
*Sender,Excascadetreelib_tlb::ViewItemUpdateEnum
Operation,Excascadetreelib_tlb::HITEM   Item)
{
}
```

Delphi

```
procedure ViewItemUpdate(ASender: TObject; Operation :
ViewItemUpdateEnum;Item : HITEM);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ViewItemUpdate(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemUpdateEvent);
begin
end;
```

Powe…

```
begin event ViewItemUpdate(long  Operation,long  Item)

end event ViewItemUpdate
```

VB.NET

```
Private Sub ViewItemUpdate(ByVal sender As System.Object, ByVal e As
AxEXCASCADETREELib._ICascadeTreeEvents_ViewItemUpdateEvent) Handles
ViewItemUpdate
End Sub
```

VB6

```
Private Sub ViewItemUpdate(ByVal Operation As
EXCASCADETREELibCtl.ViewItemUpdateEnum,ByVal Item As
EXCASCADETREELibCtl.HITEM)
End Sub
```

VBA

```
Private Sub ViewItemUpdate(ByVal Operation As Long,ByVal Item As Long)
End Sub
```

VFP
```
LPARAMETERS Operation,Item


```

Xbas…
```
PROCEDURE OnViewItemUpdate(oCascadeTree,Operation,Item)

RETURN
```

Syntax for ViewItemUpdate event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="ViewItemUpdate(Operation,Item)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ViewItemUpdate(Operation,Item)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComViewItemUpdate OLEViewItemUpdateEnum   llOperation
HITEM   llItem
   Forward Send OnComViewItemUpdate llOperation llItem
End_Procedure
```

Visual Objects
```
METHOD OCX_ViewItemUpdate(Operation,Item) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ViewItemUpdate(int   _Operation,int   _Item)
{
}
```

XBasic
```
function ViewItemUpdate as v (Operation  as
OLE::Exontrol.CascadeTree.1::ViewItemUpdateEnum,Item  as
OLE::Exontrol.CascadeTree.1::HITEM)
end function
```

dBASE
```
function nativeObject_ViewItemUpdate(Operation,Item)
return
```

event ViewStartChanging (Operation as ViewOperationEnum)

Occurs once the user is about to change the view.

Type	Description
Operation as ViewOperationEnum	A ViewOperationEnum expression that specifies the operation is about to begin.

The ViewStartChanging / ViewEndChanging events notify your application that an operation starts or ends. For instance, ViewStartChanging(exSelectionChange) / ViewEndChanging(exSelectionChange) events notify your application that an item has been selected. The View property specifies the view where the event occurs. The ViewFromPoint property retrieves the view from the point. The ViewItemFromPoint property retrieves the view and item from the point. The ViewColumnFromPoint property retrieves the view and column from the point. The ColumnFromPoint property retrieves the column from the point.

Syntax for ViewStartChanging event, **/NET** version, on:

C#
```
private void ViewStartChanging(object
sender,exontrol.EXCASCADETREELib.ViewOperationEnum  Operation)
{
}
```

VB
```
Private Sub ViewStartChanging(ByVal sender As System.Object,ByVal Operation As
exontrol.EXCASCADETREELib.ViewOperationEnum) Handles ViewStartChanging
End Sub
```

Syntax for ViewStartChanging event, **/COM** version, on:

C#
```
private void ViewStartChanging(object sender,
AxEXCASCADETREELib._ICascadeTreeEvents_ViewStartChangingEvent e)
{
}
```

C++
```
void OnViewStartChanging(long  Operation)
{
}
```

C++ Builder
```
void __fastcall ViewStartChanging(TObject
*Sender,Exmillerlib_tlb::ViewOperationEnum  Operation)
{
```

```
}
```

Delphi

```
procedure ViewStartChanging(ASender: TObject; Operation :
ViewOperationEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ViewStartChanging(sender: System.Object; e:
AxEXCASCADETREELib._ICascadeTreeEvents_ViewStartChangingEvent);
begin
end;
```

Powe…

```
begin event ViewStartChanging(long  Operation)

end event ViewStartChanging
```

VB.NET

```
Private Sub ViewStartChanging(ByVal sender As System.Object, ByVal e As
AxEXCASCADETREELib._ICascadeTreeEvents_ViewStartChangingEvent) Handles
ViewStartChanging
End Sub
```

VB6

```
Private Sub ViewStartChanging(ByVal Operation As
EXCASCADETREELibCtl.ViewOperationEnum)
End Sub
```

VBA

```
Private Sub ViewStartChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation

```

Xbas…

```
PROCEDURE OnViewStartChanging(oCascadeTree,Operation)

RETURN
```

Syntax for ViewStartChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ViewStartChanging(Operation)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ViewStartChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComViewStartChanging OLEViewOperationEnum   llOperation
    Forward Send OnComViewStartChanging llOperation
End_Procedure
```

Visual Objects

```
METHOD OCX_ViewStartChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ViewStartChanging(int   _Operation)
{
}
```

XBasic

```
function ViewStartChanging as v (Operation  as
OLE::Exontrol.ExMiller.1::ViewOperationEnum)
end function
```

dBASE

```
function nativeObject_ViewStartChanging(Operation)
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 .