

 **ExplorerTree**

Add structured navigation functionality to your applications. The ExplorerTree component adds navigation functionality to your applications, it brings simple information structuring and easy application navigation. It contains a WYSWYG designer, which is available in all environments such as .NET, VFP or else. It simplifies the organization of information in your applications. Hierarchical structure of Groups and Items allows perfect structuring of information. Create Outlook style bar and tree navigation interfaces. The ExontrolTree component combines features of the most popular [ExplorerBar](#) and [ExTree](#) components.

Features Include:

- **WYSWYG Template/Layout Editor** support
- **Skinnable Interface** support (ability to apply a skin to the any background part)
- **Shortcut bar** support (Ability to group the groups of items, in the shortcut bar).
- ADO and DAO support
- Ability to display group's items as simple tree, multi-column tree, simple list or multi-column list
- **ActiveX hosting** (you can place any ActiveX component in any item of the group)
- Events from contained components are fired through to your program using the exact same model used in VB
- 'starts with' and 'contains' **incremental searching** support
- Custom size pictures support.
- Ability to load the icons list from a BASE64 encoded string
- **FilterBar Support.** Ability to filter items with an easy-to-use interface
- **Multiple levels header** support
- Unlimited color/HTML options for cells, items
- Multiple selection
- Multi-line HTML **tooltip** support, XP shadow effect
- Background Picture Support
- Gradient colors support
- Mouse wheel support
- **Split Cells** support
- **Merge Cells** support
- **Locked** Items/Columns support
- **Divider Items** support.
- **Computed Fields** support, **Conditional Format** support
- Hyperlinks, anchor elements support.
- Radio buttons, images, check boxes
- **Partial Check Support.** Built-in checkbox reflection to reflect that state of children, parents

Classament A				
Group	P1	P2	P3	P4
Group 1				
Team 1	11	2	3	12
Team 2	2	3	4	2
Group 2				
Team 1	1	2	3	4
Team 2	5	6	6	16

Classament B				

Classament C				
Group	P1	P2	P3	P4
<input checked="" type="checkbox"/> Group 1				
<input checked="" type="checkbox"/> Team 1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Team 2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/> Group 2				
<input checked="" type="checkbox"/> Team 1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/> Team 2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

7 8 ⇅

Ž ExPlorerTree is a trademark of Exontrol. All Rights Reserved.

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 object is left aligned.
CenterAlignment	1	The object is centered.
RightAlignment	2	The object is right aligned.

constants AppearanceEnum

The AppearanceEnum enumeration is used to specify the appearance of the control's header bar. See also the [HeaderAppearance](#) property.

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 **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.

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

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

Name	Value	Description
exHeaderFilterBarButton	0	Specifies the background color for the drop down filter bar button. Use the DisplayFilterButton property to specify whether the drop down filter bar button is visible or hidden.
exFooterFilterBarButton	1	Specifies the background color for the closing button in the filter bar. Use the ClearFilter method to remove the filter from the control.
exCellButtonUp	2	Specifies the background color for the cell's button, when it is up. Use the CellHasButton property to assign a button to a cell.
exCellButtonDown	3	Specifies the background color for the cell's button, when it is down. Use the CellHasButton property to assign a button to a cell.
		Specifies the visual appearance for the header in a

exDateHeader	8	calendar control.
exDateTodayUp	9	Specifies the visual appearance for the today button in a calendar control, when it is up.
exDateTodayDown	10	Specifies the visual appearance for the today button in a calendar control, when it is down.
exDateScrollThumb	11	Specifies the visual appearance for the scrolling thumb in a calendar control.
exDateScrollRange	12	Specifies the visual appearance for the scrolling range in a calendar control.
exDateSeparatorBar	13	Specifies the visual appearance for the separator bar in a calendar control.
exDateSelect	14	Specifies the visual appearance for the selected date in a calendar control.
exSelBackColorFilter	20	Specifies the visual appearance for the selection in the drop down filter window. The drop down filter window shows up when the user clicks the filter button in the column's header. Use the DisplayFilterButton property to specify whether the drop down filter bar button is visible or hidden.
exSelForeColorFilter	21	Specifies the foreground color for the selection in the drop down filter window.
exBackColorFilter	26	Specifies the background color for the drop down filter window.
exForeColorFilter	27	Specifies the foreground color for the drop down filter window.
exSortBarLinkColor	28	Indicates the color or the visual appearance of the links between columns in the control's sort bar.
exCursorHoverColumn	32	Specifies the visual appearance for the column when the cursor hovers the column.
exDragDropBefore	33	Specifies the visual appearance for the drag and drop cursor before showing the items.
exDragDropAfter	34	Specifies the visual appearance for the drag and drop cursor after showing the items. If the exDragDropAfter option is set on white (0x00FFFFFF), the image is not showing on OLE Drag and drop.
		Specifies the graphic feedback of the item from the

exDragDropListTop

35

drag and drop cursor if the cursor is in the top half of the row. *Please note, that if a visual effect is specified for exDragDropListOver AND exDragDropListBetween states, and a visual effect is specified for exDragDropListTop OR/AND exDragDropListBottom state(s), the exDragDropListTop visual effect is displayed ONLY if the cursor is over the first visible item, and the exDragDropListBottom visual effect is shown ONLY for the last visible item. Use the [ItemFromPoint](#) property to retrieve the hit test code for the part from the cursor. This option can be changed during the OLEDragOver event to change the visual effect for the item from the cursor at runtime.*

exDragDropListBottom

36

Specifies the graphic feedback of the item from the drag and drop cursor if the cursor is in the bottom half of the row. *Please note, that if a visual effect is specified for exDragDropListOver AND exDragDropListBetween states, and a visual effect is specified for exDragDropListTop OR/AND exDragDropListBottom state(s), the exDragDropListTop visual effect is displayed ONLY if the cursor is over the first visible item, and the exDragDropListBottom visual effect is shown ONLY for the last visible item. Use the [ItemFromPoint](#) property to retrieve the hit test code for the part from the cursor. This option can be changed during the OLEDragOver event to change the visual effect for the item from the cursor at runtime.*

exDragDropForeColor

37

Specifies the foreground color for the items being dragged.

exDragDropListOver

38

Specifies the graphic feedback of the item from the cursor if it is over the item. *Please note, that if a visual effect is specified for exDragDropListOver AND exDragDropListBetween states, and a visual effect is specified for exDragDropListTop OR/AND exDragDropListBottom state(s), the exDragDropListTop visual effect is displayed ONLY if the cursor is over the first visible item, and the exDragDropListBottom visual effect is shown ONLY*

for the last visible item. Use the [ItemFromPoint](#) property to retrieve the hit test code for the part from the cursor. This option can be changed during the `OLEDragOver` event to change the visual effect for the item from the cursor at runtime.

`exDragDropListBetween` 39

Specifies the graphic feedback of the item when the drag and drop cursor is between items. *Please note, that if a visual effect is specified for `exDragDropListOver` AND `exDragDropListBetween` states, and a visual effect is specified for `exDragDropListTop` OR/AND `exDragDropListBottom` state(s), the `exDragDropListTop` visual effect is displayed ONLY if the cursor is over the first visible item, and the `exDragDropListBottom` visual effect is shown ONLY for the last visible item. Use the [ItemFromPoint](#) property to retrieve the hit test code for the part from the cursor. This option can be changed during the `OLEDragOver` event to change the visual effect for the item from the cursor at runtime.*

`exDragDropAlign`

40

Specifies the alignment of the drag and drop image relative to the cursor. By default, the `exDragDropAlign` option is 0, which initially the drag and drop image is shown centered relative to the position of the cursor.

The valid values are listed as follows (hexa representation):

- 0x00000000, (default), the drag and drop image is shown centered relative to the cursor, and shows up.
- 0x01000000, (left), the drag and drop image is shown to the left of the cursor.
- 0x02000000, (right), the drag and drop image is shown to the right of the cursor.
- 0x04000000, (center-down), the drag and drop image is shown centered relative to the cursor, and shows down.
- 0xFF000000, (as- is), the drag and drop image is shown as it is clicked.

exHeaderFilterBarActive	41	Specifies the visual appearance of the drop down filter bar button, while filter is applied to the column.
exToolTipAppearance	64	Indicates the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. se the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the CellToolTip property to specify the cell's tooltip. Use the ToolTip property to specify the group's tooltip.. Use the ShowToolTip method to display a custom tooltip.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.
exColumnsFloatBackColor	87	Specifies the background color for the Columns float bar.
exColumnsFloatScrollBackColor	88	Specifies the background color for the scroll bars in the Columns float bar.
exColumnsFloatScrollPressBackColor	89	Specifies the background color for the scroll bars in the Columns float bar, while the scroll part is pressed.
exColumnsFloatScrollUp	90	Specifies the visual appearance of the up scroll bar.
exColumnsFloatScrollDown	91	Specifies the visual appearance of the down scroll bar.
exColumnsFloatAppearance	92	Specifies the visual appearance for the frame/borders of the Column's float bar
exColumnsFloatCaptionBackColor	93	Specifies the visual appearance for caption, if the Background(exColumnsFloatAppearance) property is specified.
exColumnsFloatCaptionForeColor	94	Specifies the foreground color for the caption, if the Background(exColumnsFloatAppearance) property is specified.
exColumnsFloatCloseButton	95	Specifies the visual appearance for the closing button, if the Background(exColumnsFloatAppearance) property is specified.

exListOLEDropPosition	96	Specifies the visual appearance of the dropping position over the list part of the control, when it is implied in a OLE Drag and Drop operation.
exSelBackColorHide	166	Specifies the selection's background color, when the control has no focus.
exSelForeColorHide	167	Specifies the selection's foreground color, when the control has no focus.
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.
exVSLower	268	The lower part (exLowerBackPart) in normal state.
exVSLowerP	269	The lower part (exLowerBackPart) when it is pressed.
exVSLowerD	270	The lower part (exLowerBackPart) when it is

		disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.
exVSUpper	272	The upper part (exUpperBackPart) in normal state.
exVSUpperP	273	The upper part (exUpperBackPart) when it is pressed.
exVSUpperD	274	The upper part (exUpperBackPart) when it is disabled.
exVSUpperH	275	The upper part (exUpperBackPart) when the cursor hovers it.
exVSBack	276	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exVSBackP	277	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exVSBackD	278	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exVSBackH	279	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exHSLeft	384	The left button in normal state.
exHSLeftP	385	The left button when it is pressed.
exHSLeftD	386	The left button when it is disabled.
exHSLeftH	387	The left button when the cursor hovers it.
exHSThumb	388	The thumb part (exThumbPart) in normal state.
exHSThumbP	389	The thumb part (exThumbPart) when it is pressed.
exHSThumbD	390	The thumb part (exThumbPart) when it is disabled.
exHSThumbH	391	The thumb part (exThumbPart) when the cursor hovers it.
exHSRight	392	The right button in normal state.
exHSRightP	393	The right button when it is pressed.
exHSRightD	394	The right button when it is disabled.
exHSRightH	395	The right button when the cursor hovers it.
exHSLower	396	The lower part (exLowerBackPart) in normal state.
exHSLowerP	397	The lower part (exLowerBackPart) when it is

pressed.

exHSLowerD	398	The lower part (exLowerBackPart) when it is disabled.
exHSLowerH	399	The lower part (exLowerBackPart) when the cursor hovers it.
exHSUpper	400	The upper part (exUpperBackPart) in normal state.
exHSUpperP	401	The upper part (exUpperBackPart) when it is pressed.
exHSUpperD	402	The upper part (exUpperBackPart) when it is disabled.
exHSUpperH	403	The upper part (exUpperBackPart) when the cursor hovers it.
exHSBack	404	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exHSBackP	405	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exHSBackD	406	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exHSBackH	407	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exSBtn	324	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), in normal state.
exSBtnP	325	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is pressed.
exSBtnD	326	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is disabled.
exSBtnH	327	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when the cursor hovers it .
exScrollHoverAll	500	exScrollHoverAll. Indicates whether the hover-all feature is by default (0), always on (-1) or disabled(1).
exScrollSizeGrip	511	Specifies the visual appearance of the control's size grip when both scrollbars are shown.

constants BackModeEnum

Specifies how the control displays the selection

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

constants CaptionFormatEnum

The CaptionFormatEnum type defines how the cell/group's caption is painted

Name	Value	Description
exText	0	<p>No HTML tags are painted.</p> <p>The control uses built-in HTML tags to display the caption using HTML format. The control supports the following HTML tags:</p> <ul style="list-style-type: none">• <code> ... </code> displays the text in bold• <code><i> ... </i></code> displays the text in <i>italics</i>• <code><u> ... </u></code> <u>underlines</u> the text• <code><s> ... </s></code> Strike-through text• <code><a id;options> ... </code> 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 <code><a></code> 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 <i>AnchorClick(AnchorID, Options)</i> event when the user clicks the anchor element. The <i>FormatAnchor</i> property customizes the visual effect for anchor elements. <p>The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <code><a ;exp=></code> or <code><a ;e64=></code> anchor tags. The <code>exp/e64</code> field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.</p> <ul style="list-style-type: none">◦ <code>exp</code>, stores the plain text to be shown once the user clicks the anchor, such as "<code><a ;exp=show lines></code>"◦ <code>e64</code>, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<code><a ;e64=gA8ABmABnABjABvABshIAOQAEAA </code>" that displays show lines- in gray

when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAAC" string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.


Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** 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=rrgbb> ... </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=rrgbb> ... </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=rrgbb> ... </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=rrggb> ... </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
- **number[:width]** 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.
- **key[:width]** 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 **&**; (&), **<**; (<),

> (>), **"** (") 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 **rrggb**, **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 FFFFFFFF;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:




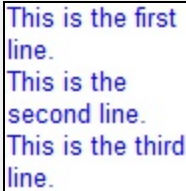
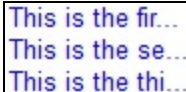
exComputedField

2

Indicates a computed field. The [CellCaption](#) or the [ComputedField](#) property indicates the formula to compute the field.

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 <code>\r\n</code> or <code>
</code> 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 <code>\r\n</code> or <code>
</code> 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 <code>\r\n</code> or <code>
</code> 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 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. <i>(Boolean expression, False)</i>
exCellHasRadioButton	1	Assigns radio buttons to all cells in the column, if it is True. Similar with the CellHasRadioButton property. <i>(Boolean expression, False)</i>
exCellHasButton	2	Specifies that all cells in the column are buttons, if it is True. Similar with the CellHasButton property. <i>(Boolean expression, False)</i>
exCellButtonAutoWidth	3	Specifies that all buttons in the column fit the cell's caption, if it is True. Similar with the CellButtonAutoWidth property. <i>(Boolean expression, False)</i>
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. <i>(Long expression)</i>
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. <i>(Long expression)</i>

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)

exCellCaptionFormat 17 exCellCaptionFormat. Specifies the type of text being displayed in the cells in the column.

Specifies the template for the column's filter when the [Filter](#) property or the 'Filter For' field is populated. This property customizes the filter pattern for the column when the [FilterType](#) property is set to exPattern. It supports the **<%filter%>**

exFilterPatternTemplate

21

keyword, which replaces the original filter input. For example, setting Def(exFilterPatternTemplate) to "[*<%filter%>*](#)" filters for all items containing the specified sequence, while setting it to "[Item*<%filter%>](#)" filters for all items starting with 'Item' and ending with the typed characters. If the Column.Def(exFilterPatternTemplate) property is empty, the filter is applied as it is (no effect).

(*String expression*)

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")

constants DescriptionTypeEnum

The group's [Description](#) property defines descriptions for few group parts.

Name	Value	Description
exFilterBarAll	0	Defines the caption of (All) in the filter bar window.
exFilterBarBlanks	1	Defines the caption of (Blanks) in the filter bar window.
exFilterBarNonBlanks	2	Defines the caption of (NonBlanks) in the filter bar window.
exFilterBarFilterForCaption	3	Defines the caption of "Filter For:" in the filter bar window.
exFilterBarFilterTitle	4	Defines the title for the filter tooltip.
exFilterBarPatternFilterTitle	5	Defines the title for the filter pattern tooltip.
exFilterBarTooltip	6	Defines the tooltip for filter window.
exFilterBarPatternTooltip	7	Defines the tooltip for filter pattern window.
exFilterBarFilterForTooltip	8	Defines the tooltip for "Filter For:" window.
exFilterBarIsBlank	9	Defines the caption of the function 'IsBlank' in the control's filter bar.
exFilterBarIsNonBlank	10	Defines the caption of the function 'not IsBlank' in the control's filter bar.
exFilterBarAnd	11	Customizes the ' and ' text in the control's filter bar when multiple columns are used to filter the items in the control.
exFilterBarDate	12	Specifies the "Date:" caption being displayed in the drop down filter window when DisplayFilterPattern property is True, and DisplayFilterDate property is True.
exFilterBarDateTo	13	Specifies the "to" sequence being used to split the from date and to date in the Date field of the drop down filter window. For instance, the "to 12/13/2004" specifies the items before 12/13/2004, "12/23/2004 to 12/24/2004" filters the items between 12/23/2004 and 12/24/2004, or "Feb 12 2004 to" specifies all items after a date.
		Describes the tooltip that shows up when cursor is over the Date field. "You can filter the items into a given interval of dates. For instance, you can filter

exFilterBarDateTooltip 14 all items dated before a specified date (**to 2/13/2004**), or all items dated after a date (**Feb 13 2004 to**) or all items that are in a given interval (**2/13/2004 to 2/13/2005**)."

exFilterBarDateTitle 15 Describes the title of the tooltip that shows up when the cursor is over the Date field. By default, the exFilterBarDateTitle is "Date".

exFilterBarDateTodayCaption 16 Specifies the caption for the 'Today' button in a date filter window. By default, the exFilterBarDateTodayCaption property is "Today".

exFilterBarDateMonths 17 Specifies the name for months to be displayed in a date filter window. The list of months should be delimited by space characters. By default, the exFilterBarDateMonths is "January February March April May June July August September October November December".

exFilterBarDateWeekDays 18 Specifies the shortcut for the weekdays to be displayed in a date filter window. The list of shortcut for the weekdays should be separated by space characters. By default, the exFilterBarDateWeekDays is "S M T W T F S". The first shortcut in the list indicates the shortcut for the Sunday, the second shortcut indicates the shortcut for Monday, and so on.

exFilterBarChecked 19 Defines the caption of (Checked) in the filter bar window. The exFilterBarChecked option is displayed only if the [FilterType](#) property is exCheck. If the Description(exFilterBarChecked) property is empty, the (Checked) predefined item is not shown in the drop down filter window. If the user selects the (Checked) item the control filter checked items. The [CellState](#) property indicates the state of the cell's checkbox.

exFilterBarUnchecked 20 Defines the caption of (Unchecked) in the filter bar window. The exFilterBarUnchecked option is displayed only if the [FilterType](#) property is exCheck. If the Description(exFilterBarUnchecked) property is empty, the (Unchecked) predefined item is not shown in the drop down filter window. If the user selects the (Unchecked) item the control filter unchecked items. The [CellState](#) property indicates

the state of the cell's checkbox.

exFilterBarIsChecked	21	Defines the caption of the 'IsChecked' function in the control's filter bar. The 'IsChecked' function may appear only if the user selects (Checked) item in the drop down filter window, when the FilterType property is exCheck.
exFilterBarIsUnchecked	22	Defines the caption of the 'not IsChecked' function in the control's filter bar. The 'not IsChecked' function may appear only if the user selects (Unchecked) item in the drop down filter window, when the FilterType property is exCheck.
exFilterBarOr	23	Customizes the 'or' operator in the control's filter bar when multiple columns are used to filter the items in the control.
exFilterBarNot	24	Customizes the 'not' operator in the control's filter bar
exFilterBarExclude	25	Specifies the 'Exclude' caption being displayed in the drop down filter.
exColumnsFloatBar	26	Specifies the caption to be shown on control's Columns float bar.

constants DividerAlignmentEnum

Defines the alignment for a divider line into 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.

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 exClipboardFormatEnum

Defines the clipboard format constants. Use [GetFormat](#) property to check whether the clipboard data is of given type

Name	Value	Description
exCFText	1	Null-terminated, plain ANSI text in a global memory bloc.
exCFBitmap	2	A bitmap compatible with Windows 2.x.
exCFMetafile	3	A Windows metafile with some additional information about how the metafile should be displayed.
exCFDIB	8	A global memory block containing a Windows device-independent bitmap (DIB).
exCFPalette	9	A color-palette handle.
exCFEMetafile	14	A Windows enhanced metafile.
exCFFiles	15	A collection of files. Use Files property to get or set the collection of files.
exCFRTF	-16639A	RTF document.

constants exOLEDragOverEnum

State transition constants for the OLEDragOver event

Name	Value	Description
exOLEDragEnter	0	Source component is being dragged within the range of a target.
exOLEDragLeave	1	Source component is being dragged out of the range of a target.
exOLEDragOver	2	Source component has moved from one position in the target to another.

constants exOLEDDropEffectEnum

Drop effect constants for OLE drag and drop events.

Name	Value	Description
exOLEDDropEffectNone	0	Drop target cannot accept the data, or the drop operation was cancelled.
exOLEDDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
exOLEDDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
exOLEDDropEffectScroll	-2147483648	This one is not implemented.

constants exOLEDropModeEnum

Constants for the OLEDropMode property, that defines how the control accepts OLE drag and drop operations. Use the [OLEDropMode](#) property to set how the component handles drop operations.

Name	Value	Description
exOLEDropNone	0	The control is not used OLE drag and drop functionality.
exOLEDropManual	1	The control triggers the OLE drop events, allowing the programmer to handle the OLE drop operation in code.

Here's the list of events related to OLE drag and drop: [OLECompleteDrag](#), [OLEDragDrop](#), [OLEDragOver](#), [OLEGiveFeedback](#), [OLESetData](#), [OLEStartDrag](#).

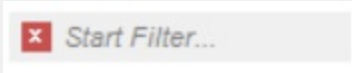


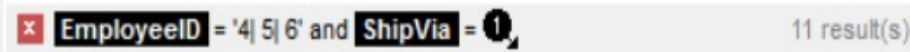
constants ExpandButtonEnum

Defines how the group displays the expanding/collapsing buttons.

Name	Value	Description
exNoButtons	0	The group displays no expand buttons.
exPlus	-1	A plus sign is displayed for collapsed items, and a minus sign for expanded items. (⊕ ⊖)
exArrow	1	The group uses icons to display the expand buttons. (▶ ▼)
exCircle	2	The group uses icons to display the expand buttons. (⊕ ⊖)
exWPlus	3	The group uses icons to display the expand buttons. (⊕ ⊖)
exCustom	4	The HasButtonsCustom property specifies the index of icons being used for +/- signs on parent items.

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 through 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

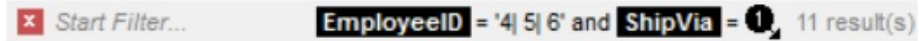
The exFilterBarShowCloseOnRight flag specifies that the close button of the control's filter bar should be displayed on the right side. If the control's [RightToLeft](#) property is True, the close button of the control's filter bar would be automatically displayed on the left side.



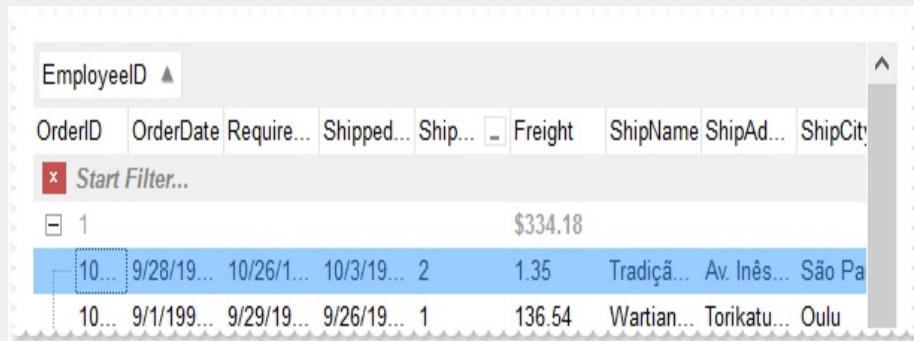
exFilterBarCompact

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.



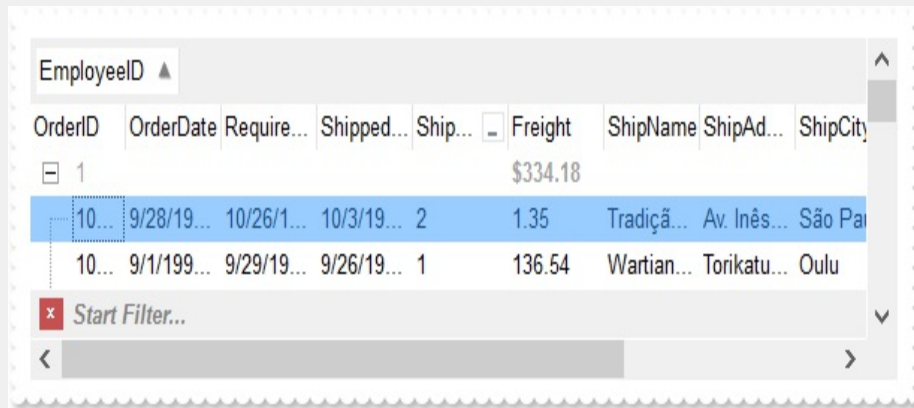
The exFilterBarTop flag displays the filter-bar on top (between control's header and items section as shown):



exFilterBarTop

8192

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 below:

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

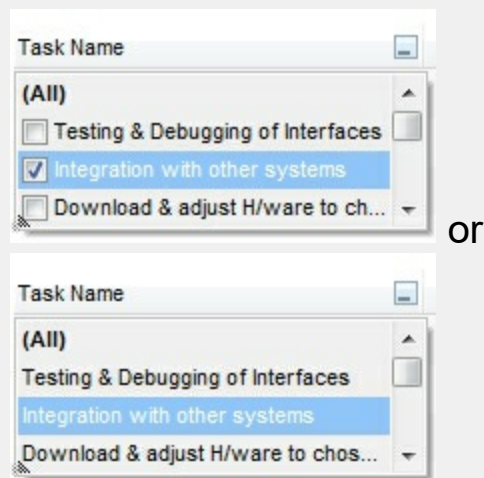
selection in the drop down filter list.

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.

exShowCheckBox

256

The following screen shot shows the drop down filter **with** or **with no** exShowCheckBox flag:

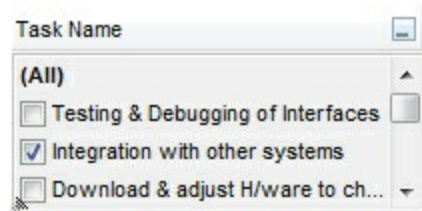


The selection background is not shown for checked items in the filter's list. This flag can be combined with exShowCheckBox.

exHideCheckSelect

512

The following screen shot shows no selection background for the checked items:



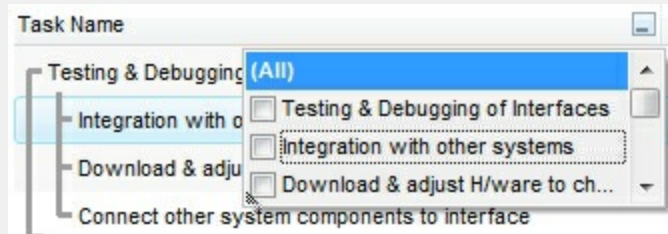
This flag allows highlighting the focus cell value in the filter's list. The focus cell value is the cell's content at the moment the drop down filter window is shown. For instance, click an item so a new item is selected, and click the drop down filter button. A

item being focused in the drop down filter list is the one you have in the control's selection. This flag has effect also, if displaying a calendar control in the drop down filter list.

exShowFocusItem

1024

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

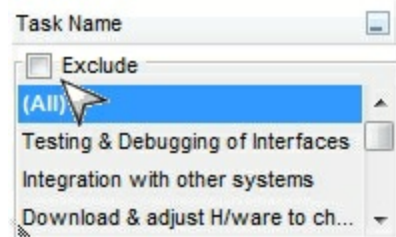
This flag indicates whether the filter's tooltip is shown. The [Description](#)(exFilterBarToolTip,exFilterBarPatternTool...) properties defines the filter's tooltips.

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. The FilterBarPromptPattern property may include wild characters as follows: <ul style="list-style-type: none">• '?' 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 or exFilterPromptEndWith.

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

The FilterTypeEnum type defines the type of filter applies to a column. Use the [FilterType](#) property to specify the type of filter being used. Use the [Filter](#) property to specify the filter being used. The value for [Filter](#) property depends on the [FilterType](#) property. Use the [Description](#) property to customize the captions for control filter bar window. The [FilterList](#) property indicates the values the drop-down filter includes. The FilterTypeEnum type supports the following values:

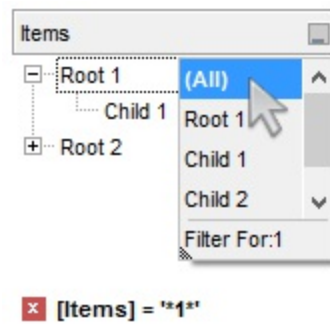
Name

Value Description

exAll

0

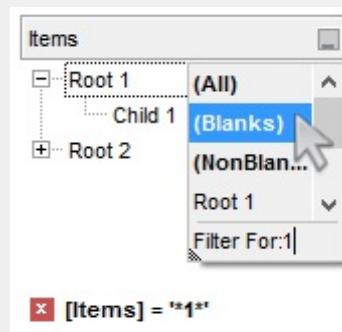
No filter applied. Use the [Description](#) property to change the "(All)" caption in the drop down filter.



exBlanks

1

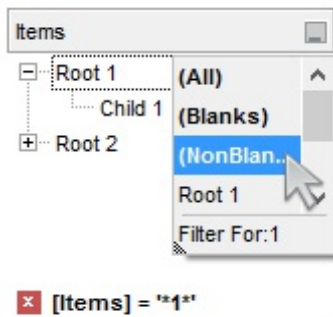
Only blank items are included. Use the [Description](#) property to change the "(Blanks)" caption in the drop down filter. The [Filter](#) property has no effect.



exNonBlanks

2

Only non blanks items are included. Use the [Description](#) property to change the "(NonBlanks) " caption in the drop down filter. The [Filter](#) property has no effect.



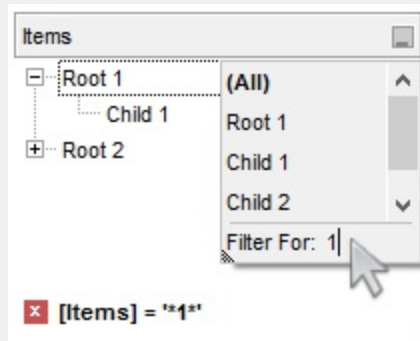
exPattern

3

Only items that match the pattern are included. The [Filter](#) property 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, and [chars] indicates a group of characters. If any of the *, ?, # or | characters are preceded by a \ (escape character) it masks the character itself. The [Def\(exFilterPatternTemplate\)](#) property specifies the template for the column's filter when the [Filter](#) property or the 'Filter For' field is populated. The exFilterDoCaseSensitive flag can be combined with exPattern or exFilter types, indicating that case-sensitive filtering should be performed.

For instance:

- **"*1"**, only items that ends with 1 are included
- **"A*|B*"**, only items that starts with a/A or b/B



Only items (of date type) within the specified range are included. The [Filter](#) property defines the interval of dates being used to filter items. The interval of dates should be as [dateFrom] to [dateTo]. Use the [Description](#) property to change the "to" conjunction used to split the dates in the interval. If the dateFrom value is missing, the control includes only

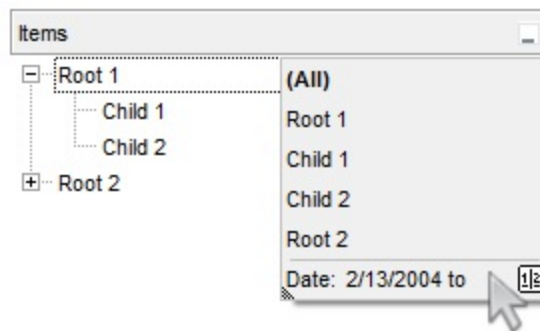
the items before the dateTo date, if the dateTo value is missing, the control includes the items after the dateFrom date. If both dates (dateFrom and dateTo) are present, the control includes the items between this interval of dates. The [DisplayFilterDate](#) property specifies whether the drop down filter window displays a date selector to specify the interval dates to filter for.

exDate

4

For instance:

- "2/13/2004 to" includes all items after 2/13/2004 inclusive
- "2/13/2004 to Feb 14 2005" includes all items between 2/13/2004 and 2/14/2004



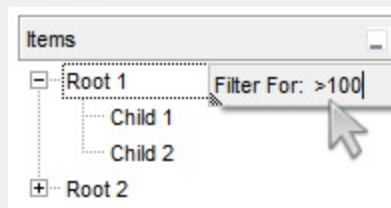
Only items (of numeric type) within the specified range are included. The [Filter](#) property may include operators like <, <=, =, <>, >= or > and numbers to define rules to include numbers in the control's list. If the FilterType property is exNumeric, the drop down filter window doesn't display the filter list that includes items "(All)", "(Blanks)", ... and so on.

exNumeric

5

For instance:

- "100", filter items with the value 100
- "> 10 < 100", indicates all numbers greater than 10 and less than 100



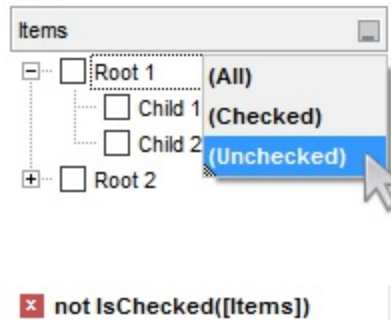
Only checked or unchecked items are included. The [CellState](#) property indicates the state of the cell's checkbox. The [Filter](#) property on "0" filters for unchecked items, while "1" filters for checked items. A checked item has the the CellState property different than zero. An unchecked item has the CellState property on zero.

For instance:

exCheck

6

- "0", only unchecked items are included
- "1", only checked items are included



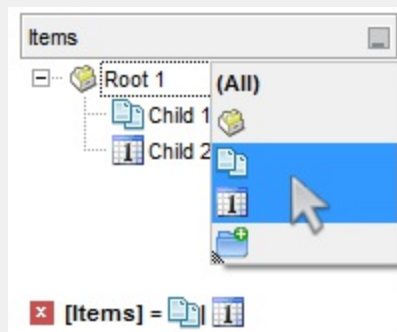
Only items showing the specified icons (icon index) are included. The [CellImage](#) property indicates the cell's icon. Multiple icons are separated by the '|' character. The [Filter](#) property defines the list of icons, separated by the '|' character, to apply the filter.

For instance:

exImage

10

- "1", only items that displays the icons with the index 1 are included
- "2|3", only items displaying the icons with index 2 or 3 are included



Only the items that are in the [Filter](#) property are

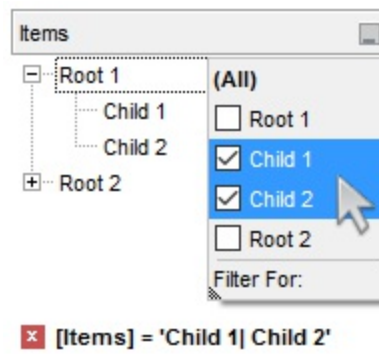
included. Multiple items are separated by the '|' character. The exShowCheckBox flag of [FilterList](#) property displays a check box for each included item. The exFilterDoCaseSensitive flag can be combined with exPattern or exFilter types, indicating that case-sensitive filtering should be performed.

For instance:

exFilter

240

- "Item 1", only items with the caption 'Item 1' are included
- "Item 3|Item 3", only items displaying icons with an index of 2 or 3 are included



exFilterDoCaseSensitive

256

If this flag is present, the column filtering is case-sensitive. If this flag is missing, the filtering is case-insensitive by default. The exFilterDoCaseSensitive flag can be used to enable case-sensitive filtering within the column. However, this flag is not applied to the filter prompt feature. The exFilterDoCaseSensitive flag can be combined with exPattern or exFilter types.

The flag indicates that the Exclude field of the column is checked, meaning items that match the filter are excluded from the list. The exShowExclude flag of [FilterList](#) property indicates whether the Exclude option is shown in the drop down filter window.

exFilterExclude

512

Items

- [-] Root 1
 - Child 2
- [+] Root 2

Exclude

(All)

- Root 1
- Child 1
- Child 2
- Root 2

[Items] # 'Child 1'

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 group paints the grid lines.

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

constants GridLineStyleEnum

The GridLineStyle 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 GridLineStyle 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 group paints the hierarchy lines.

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

constants HitTestInfoEnum

The HitTestInfoEnum expression defines the hit area within a cell. Use the [ItemFromPoint](#) 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.
exHTCellIndent	2	In the indentation associated with a cell.
exHTCellInside	4	On the icon, picture, check or caption associated with a cell.
exHTCellCaption	20	In the caption associated with a cell.
exHTCellCheck	36	In the check/radio button associated with a cell.
exHTCellIcon	68	In first icon associated with a cell.
exHTCellPicture	132	In a picture associated to a cell.
exHTCellCaptionIcon	1044	In the icon's area inside the cell's caption.
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	(HEXA 1000) 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 two items using HitTestCode AND 0x1000

constants LinesAtRootEnum

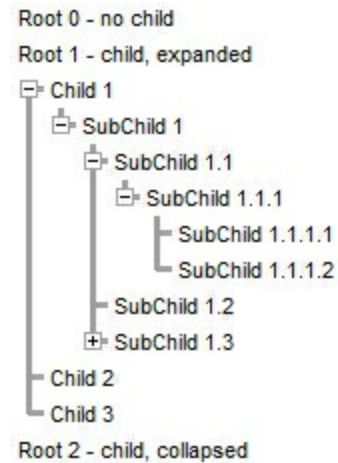
Defines how the control displays the lines at root. The LinesAtRoot property defines the way the tree lines are shown. The HasLines property defines the type of the line to be shown. The HasButtons property defines the expand/collapse buttons for parent items.

The LinesAtRootEnum type support the following values:

Name	Value	Description
		No lines at root items.

exNoLinesAtRoot

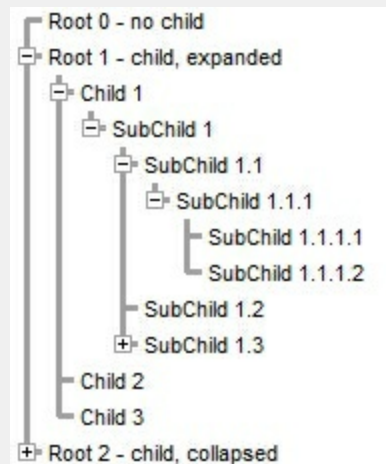
0



The control links the root items.

exLinesAtRoot

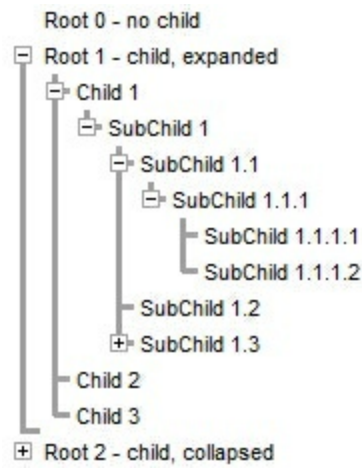
-1



The control shows no links between roots, and divides them as being in the same group.

exGroupLinesAtRoot

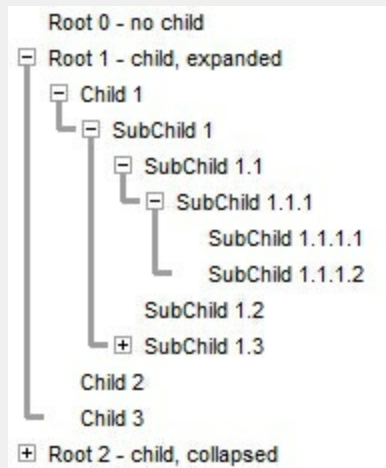
1



The lines between root items are no shown, and the links show the items being included in the group.

exGroupLines

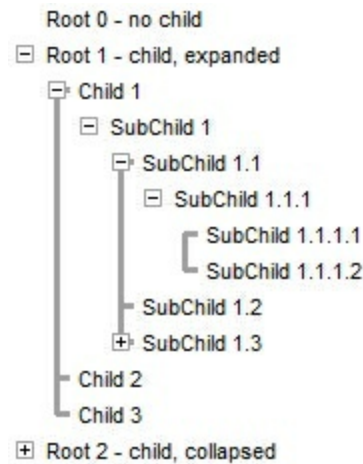
2



The lines between root items are no shown, and the links are shown between child only.

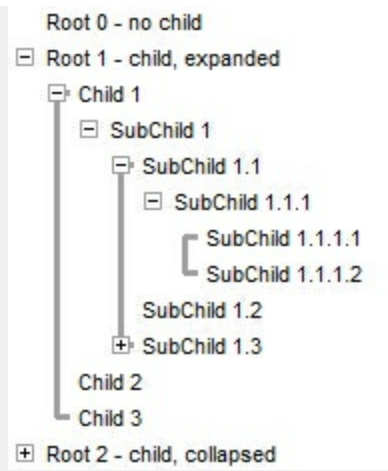
exGroupLinesInside

3



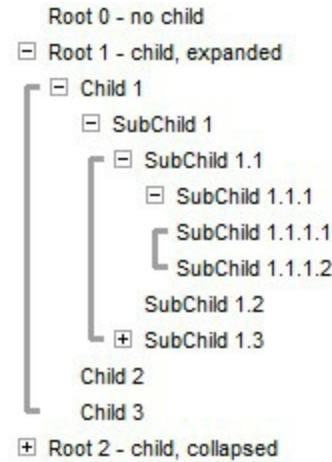
The lines between root items are no shown, and the links are shown for first and last visible child item.

exGroupLinesInsideLeaf 4



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.

exGroupLinesOutside 5



constants `PictureDisplayEnum`

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

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

constants ScrollBarEnum

The ScrollBarEnum type specifies the vertical or horizontal scroll bar in the control. Use the [ScrollBars](#) property to specify whether the vertical or horizontal scroll bar is visible or hidden. 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.
exHScroll	1	Indicates the horizontal scroll bar.

constants ScrollBarsEnum

Specifies which scroll bars will be visible on a group.

Name	Value	Description
exNoScroll	0	NoScroll. No scroll bars are shown
exHorizontal	1	Horizontal. Only horizontal scroll bars are shown.
exVertical	2	Vertical. Only vertical scroll bars are shown.
exBoth	3	Both. Both horizontal and vertical scroll bars are shown.

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. The control fires the [ScrollButtonClick](#) event when the user clicks any button in the control's scrollbar.



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

exRightB2Part	16	(R2) The second additional button in the right or down side. By default, this button is hidden.
exRightB3Part	8	(R3) The third additional button in the right or down side. By default, this button is hidden.
exRightB4Part	4	(R4) The forth additional button in the right or down side. By default, this button is hidden
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 group takes when user clicks the column's header. The [SortOnClick](#) Property specifies whether the group sorts a column when its caption is 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 sort order. 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 ways how the control can sort the columns. Use the [SortType](#) property to specify how the column gets sorted. The [CellCaption](#) property indicates the values being sorted.

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 CellData property indicates the values being sorted. Values are sorted as numbers.
SortUserDataString	6	The CellData property indicates the values being sorted. Values are sorted as strings.
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 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 UVisualThemeEnum

The UVisualThemeEnum expression specifies the UI parts that the control can shown using the current visual theme. The [UseVisualTheme](#) property specifies whether the UI parts of the control are displayed using the current visual theme.

Name	Value	Description
exNoVisualTheme	0	exNoVisualTheme
exDefaultVisualTheme	16777215	exDefaultVisualTheme
exHeaderVisualTheme	1	exHeaderVisualTheme
exFilterBarVisualTheme	2	exFilterBarVisualTheme
exButtonsVisualTheme	4	exButtonsVisualTheme
exCalendarVisualTheme	8	exCalendarVisualTheme
exCheckBoxVisualTheme	64	exCheckBoxVisualTheme
explorerBarVisualTheme	512	explorerBarVisualTheme

constants VAlignmentEnum

Specifies how the cell's caption is vertically aligned.

Name	Value	Description
TopAlignment	0	The caption is aligned to top of the cell.
MiddleAlignment	1	The cell's caption is vertically centered.
BottomAlignment	2	The caption is aligned to bottom of the cell.

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.

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

Adds or replaces a skin object to the control.

Type	Description
ID as Long	<p>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.</p>
Skin as Variant	<p>A string expression that indicates:</p> <ul style="list-style-type: none">• an Windows XP Theme part, it should start with "XP:". 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. In this case the format of the Skin parameter should be: "XP: Control/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 like listed at the end of the document. This option is available only on Windows XP that supports Themes API.• copy of another skin with different coordinates, if it begins with "CP:" . For instance, you may need to display a specified skin on a smaller rectangle. In this case, the string starts with "CP:", and contains the following "<u>CP:n l t r b</u>", where the n is the identifier being copied, the l, t, r, and b indicate the left, top, right and bottom coordinates being used to adjust the rectangle where the skin is displayed.• the path to the skin file (*.ebn). The Exontrol's exButton component installs a skin builder that should be used to create new skins• the BASE64 encoded string that holds a skin file (*.ebn). Use the Exontrol's exImages tool to build BASE 64 encoded strings on the skin file (*.ebn) you have created. Loading the skin from a file (eventually uncompressed file) is always faster then loading from a BASE64 encoded string <p>A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. You can use this</p>

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 of 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)).

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 its 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.




The identifier you choose for the skin is very important to be used in the background properties like explained below. Shortly, the color properties uses 4 bytes (DWORD, double WORD, and so on) to hold a RGB value. More than that, the first byte (most significant byte in the color) is used only to specify system color. if the first bit in the byte is 1, the rest of bits indicates the index of the system color being used. So, we use the last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used.

So, since the 7 bits can cover 127 values, excluding 0, we have 126 possibilities to store an identifier in that byte. This way, a DWORD expression indicates the background color stored in RRGGBB format and the index of the skin (ID parameter) in the last 7 bits in the high significant byte of the color. For instance, the BackColor = BackColor Or &H2000000 indicates that we apply the skin with the index 2 using the old color, to the object that BackColor is applied.

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

- control's borders, [Appearance](#) property
- control's **header bar**, [BackColorHeader](#) property
- control's **filter bar**, [FilterBarBackColor](#) property
- **selected item** or cell, [SelBackColor](#) property
- **item**, [ItemBackColor](#) property
- **cell**, [CellBackColor](#) property
- cell's **button**, "drop down" filter bar button, "close" filter bar button, tooltips, and so on, [Background](#) property
- and so on.

The following VB sample changes the visual appearance for group headers. The [BackColorGroup](#) property indicates the default group's background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the BackColorGroup property to indicates the index of the skin that we want to use. The sample applies the "" to the headers and the control looks like follows:



```
With ExplorerTree1
```

```
.VisualAppearance.Add 1, "D:\Temp\ExplorerTree.Help\tabup2.ebn"
```

```
.BackColorGroup = &H1000000
```

```
End With
```

The following C++ sample changes the visual appearance for group headers:

```
#include "Appearance.h"
```

```
m_explorertree.GetVisualAppearance().Add( 1, COleVariant(  
"D:\\Temp\\ExplorerTree.Help\\tabup2.ebn" ) );
```

```
m_explorertree.SetBackColorGroup( 0x1000000 );
```

The following VB.NET sample changes the visual appearance for group headers:

```
With AxExplorerTree1
```

```
.VisualAppearance.Add(1, "D:\Temp\ExplorerTree.Help\tabup2.ebn")
```

```
.Template = "BackColorGroup = 16777216"
```

```
End With
```

The following C# sample changes the visual appearance for group headers:

```
axExplorerTree1.VisualAppearance.Add(1, "D:\\Temp\\ExplorerTree.Help\\tabup2.ebn");
```

```
axExplorerTree1.Template = "BackColorGroup = 16777216";
```

The following VFP sample changes the visual appearance for group headers:

```
With thisform.ExplorerTree1
```

```
.VisualAppearance.Add(1, "D:\Temp\ExplorerTree.Help\abup2.ebn")
```

```
.BackColorGroup = 16777216
```

```
EndWith
```

where the 16777216 value represents 0x1000000 in hexadecimal.

The [screen shot](#) was generated using the following template:

```
BeginUpdate
```

```
Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUlK8plUrlktl
```

```
Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUlK8plUrlktl
```

Images("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0pIUrlkt

VisualAppearance

{

' Header

Add(1,"gBFLBCJwBAEHhEJAEGg4BawDg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAAGEYxR

' HeaderFilterBarButton

Add(2,"gBFLBCJwBAEHhEJAEGg4BAQEg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAAGEYxR

Add(3,"gBFLBCJwBAEHhEJAEGg4BBAEg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAAGEYxR

' SelectedItem

Add(4,

"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAYDIKsEQGGIZRhhGlwAgaFIXQK

' Marks a cell

Add(5,"gBFLBCJwBAEHhEJAEGg4BEcMQAAYAQGKIYBkAKBQAGaAoDDMOILQiMQxDPBMK

Add(6,"gBFLBCJwBAEHhEJAEGg4BaAFg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA

Add(10,

"gBFLBCJwBAEHhEJAEGg4BRIGg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUAIIRZGM

}

Background(0) = 33554432 '0x02BBGGRR

Background(1) = 50331648 '0x03BBGGRR

Background(8) = 67108864 '0x04BBGGRR

Background(9) = 67108864 '0x04BBGGRR

Background(10) = 100663296 '0x06BBGGRR
Background(11) = 100663296 '0x06BBGGRR
Background(12) = 100663296 '0x06BBGGRR
Background(13) = 100663296 '0x06BBGGRR
Background(14) = 100663296 '0x06BBGGRR

BackColor = RGB(255,255,255)
BackColorGroup = 167772160
BackColorGroup2 = RGB(198,207,247)

GroupAppearance = 7

ToolTipWidth = 154

GroupHeight = 36

Groups

```
{  
  Add("  
  Calendar ActiveX ( MSCAL.Calendar ) ")
```

```
  {  
    CaptionFormat = 1  
    Alignment = 0  
    Image =  
    "gBHJJGHA5MIgAEIe4AAAFhwQiAbCABigbEsWGAIGA7Eo7HcbIowlpFHZQkZQKA7IspIErIBt
```

```
    Height = 168
```

```
    IndentGroupLeft = 18
```

```
    Items
```

```
    {  
      Dim h  
      h = InsertControlItem("MSCAL.Calendar")
```

```
      ItemHeight( h ) = 168
```

```
      ItemObject(h)
```

```
      {  
        BackColor = RGB(198,207,247)
```

```
      }
```

```
    }
```

```
  }
```

```
  "
```

```
Mail"
```

```
{
  CaptionFormat = 1
  Expanded = True
  Alignment = 0
  Image =
"gBHJJGHA5MliAEIe4AAAFhwFIJpApWPoFNSbCAPB4QJhLCBWKoQNRpCB+V8IXIQRDDDC
```

```
  IndentGroupLeft = 18
```

```
  AutoHeight = True
```

```
  BeginUpdate
```

```
    ExpandOnSearch = True
```

```
    BackColorList = RGB(255,255,255)
```

```
    BorderColor = RGB(198,207,247)
```

```
    Indent = 12
```

```
    HasLines = -1
```

```
    HasButtons = 2
```

```
    LinesAtRoot = -1
```

```
    FullRowSelect = False
```

```
    'SelBackColor = RGB(198,207,247)
```

```
    SelBackColor = 67108864          '0x04BBGGRR
```

```
    SelForeColor = RGB(0,0,0)
```

```
    ShowFocusRect = False
```

```
  Columns
```

```
  {
```

```
    Item(0)
```

```
    {
```

```
      Caption = "Organizer"
```

```
    }
```

```
  }
```

```
  Items
```

```
  {
```

```
    Dim h, h1, h2
```

```
    h = AddItem("Inbox (23)")
```

```
    CellCaptionFormat(h,0) = 1
```

```
    CellImages(h,0) = "1,2"
```

CellToolTip(h,0) = "An **Inbox** is a file in which mail is delivered by the operating system.

Exontrol's **ExplorerBar**"

CellPicture(h,0) =

"gBHJJGHA5MIqAAXAD3AENhozhpmhqZhrMhr/h0QGcQM0QTMQZkQf8QAESGcSM0STM

ItemHeight(h) = 26

h1 = InsertItem(h, "**Unread Email** (11)")

CellCaptionFormat(h1,0) = 1

CellImage(h1,0) = 2

h1 = InsertItem(h, "**To Follow Up** (7)")

CellCaptionFormat(h1,0) = 1

CellImage(h1,0) = 3

h2 = InsertItem(h, "My Folders")

CellImage(h2,0) = 2

h1 = InsertItem(h2, "Info (128)")

CellImage(h1,0) = 2

h1 = InsertItem(h2, "Personal (59)")

CellImage(h1,0) = 3

h1 = InsertItem(h2, "Programming (159)")

CellImage(h1,0) = 1

h1 = InsertItem(h, "Draft (0)")

CellCaptionFormat(h1,0) = 1

CellImage(h1,0) = 1

h1 = InsertItem(h, "OutBox")

CellImage(h1,0) = 2

h1 = InsertItem(h, "SentItems")

CellImage(h1,0) = 3

h1 = InsertItem(h, "Deleted Items")

CellImage(h1,0) = 4

}

EndUpdate

}

Add("

Contacts **(6)**")

```
{
  Expanded = True
  CaptionFormat = 1
  Alignment = 0
  AutoHeight = True
  Image =
"gbHJJGHA5MIgAEIe4AAAFhwQiAbCAFDcVEoICEXEowjg7GAbHY7CEhHZFkRFIBQIoQKEtLZ

  IndentGroupLeft = 18
  Expanded = True
  BackColorList = RGB(255,255,255)
  BorderColor = RGB(198,207,247)
  BeginUpdate
  HeaderVisible = True
  HeaderAppearance = 6
  BackColorHeader = RGB(198,207,247)
  FilterBarBackColor = 16777216      '0x01BBGGRR
  FilterBarForeColor = RGB(255,255,255)
  BackColorHeader = 16777216      '0x01BBGGRR
  ForeColorHeader = RGB(255,255,255)
  BackColorLevelHeader = RGB(255,255,255)
  DrawGridLines = -1
  SelBackColor = RGB(198,207,247)
  SelBackColor = 67108864          '0x04BBGGRR
  SelForeColor = RGB(0,0,0)
  ShowFocusRect = False
  MarkSearchColumn = False
  AllowEdit = True
  TreeColumnIndex = -1
  Columns
  {
    Item(0)
    {
      Caption = "Name"
      Width = 130
      DisplayFilterButton = True
      DisplayFilterDate = True
    }
  }
}
```

```
}
Add("Phone")
{
Width = 100
}
1
{
  AllowSizing = False
  HTMLCaption = "1 First"
  Def(0) = True
  LevelKey = 1
  Width = 25
  Alignment = 1
}
2
{
  AllowSizing = False
  HTMLCaption = "2 Second"
  Def(0) = True
  LevelKey = 1
  Width = 25
  Alignment = 1
}
3
{
  AllowSizing = False
  HTMLCaption = "3 Third"
  Def(0) = True
  LevelKey = 1
  Width = 25
  PartialCheck = True
  Alignment = 1
}
...
{
  LevelKey = 1
  Width = 20
```



```
}
```

```
}
```

```
Items
```

```
{
```

```
Dim h
```

```
h = AddItem("Mihai Filimon")
```

```
CellCaptionFormat(h,0) = 1
```

```
CellCaption(h,1) = "744-845287"
```

```
h = AddItem("Dean Thomas")
```

```
CellCaptionFormat(h,0) = 1
```

```
CellCaption(h,1) = "928-120203"
```

```
h = AddItem("Dave Nichols")
```

```
CellCaptionFormat(h,0) = 1
```

```
CellMerge(h,1) = 2
```

```
CellCaption(h,1) = "121-121901"
```

```
h = AddItem("Brian Thompson")
```

```
CellBackColor(h,1) = 83886080
```

```
CellCaptionFormat(h,0) = 1
```

```
CellCaption(h,1) = "234-129011"
```

```
h = AddItem("Alex Antolini")
```

```
CellCaptionFormat(h,0) = 1
```

```
CellCaption(h,1) = "234-12112"
```

```
CellHAlignment(h,0) = 1
```

```
CellMerge(h,0) = 1
```

```
SortChildren(,0)
```

```
}
```

```
EndUpdate
```

```
}
```

```
Add("
```

```
Tasks")
```

```
{
```

```
CaptionFormat = 1
```

```
BackColorList = RGB(255,255,255)
```

```
BorderColor = RGB(198,207,247)
```

```
IndentGroupLeft = 18
```

```

Alignment = 0
Image =
"gBHJJGHA5MIqAAXAD3AENhozhpmhqZhrMhr/h0QGcQM0QTMQZkQf8QAESGcSM0STM

Expanded = True
AutoHeight = True
BeginUpdate
TreeColumnIndex = -1
FullRowSelect = False
SelBackColor = RGB(198,207,247)
SelForeColor = RGB(0,0,0)
SelBackColor = 67108864      '0x04BBGGRR
SelForeColor = RGB(0,0,0)
ShowFocusRect = False
Columns
{
  Item(0)
  {
    Alignment = 1
  }
}
Items
{
  Dim h
  h = AddItem("no tasks")
}
EndUpdate
}
}
EndUpdate

```

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

Control/ClassName	Part	States
		CBS_UNCHECKED
		1 CBS_UNCHECKE
		CBS_UNCHECKED

BUTTON	BP_CHECKBOX = 3	= 3 CBS_UNCHECKED = 4 CBS_CHECKED 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRES 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_CHECKED 5 RBS_CHECKEDH RBS_CHECKEDPR RBS_CHECKEDDIS
	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1 CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED :
COMBOBOX	CP_DROPDOWNBUTTON = 1	
EDIT	EP_CARET = 2	ETS_NORMAL = 1 2 ETS_SELECTED ETS_DISABLED = ETS_FOCUSED = ETS_READONLY = ETS_ASSIST = 7
	EP_EDITTEXT = 1	
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	EBHC_NORMAL = EBHC_HOT = 2
	EBP_HEADERCLOSE = 2	

EBP_HEADERPIN = 3

EBP_IEBARMENU = 4

EBP_NORMALGROUPBACKGROUND = 5

EBP_NORMALGROUPCOLLAPSE = 6

EBP_NORMALGROUPEXPAND = 7

EBP_NORMALGROUPHEAD = 8

EBP_SPECIALGROUPBACKGROUND = 9

EBP_SPECIALGROUPCOLLAPSE = 10

EBP_SPECIALGROUPEXPAND = 11

EBP_SPECIALGROUPHEAD = 12

HEADER

HP_HEADERITEM = 1

HP_HEADERITEMLEFT = 2

HP_HEADERITEMRIGHT = 3

HP_HEADERSORTARROW = 4

LISTVIEW

LVP_EMPTYTEXT = 5

LVP_LISTDETAIL = 3

LVP_LISTGROUP = 2

LVP_LISTITEM = 1

EBHC_PRESSED =

EBHP_NORMAL =

EBHP_HOT = 2

EBHP_PRESSED =

EBHP_SELECTED =

4 EBHP_SELECTED =

EBHP_SELECTED =

6

EBM_NORMAL = 1

= 2 EBM_PRESSED =

EBNGC_NORMAL :

EBNGC_HOT = 2

EBNGC_PRESSED :

EBNGE_NORMAL :

EBNGE_HOT = 2

EBNGE_PRESSED :

EBSGC_NORMAL :

EBSGC_HOT = 2

EBSGC_PRESSED :

EBSGE_NORMAL :

EBSGE_HOT = 2

EBSGE_PRESSED :

HIS_NORMAL = 1

2 HIS_PRESSED =

HILS_NORMAL = 1

= 2 HILS_PRESSED =

HIRS_NORMAL = 1

= 2 HIRS_PRESSED =

HSAS_SORTEDUP :

HSAS_SORTEDDC :

LIS_NORMAL = 1

2 LIS_SELECTED :

LIS_DISABLED = 4

LIS_SELECTEDNO
5

LVP_LISTSORTEDDETAIL = 4

MENU

MP_MENUBARDROPDOWN = 4

MP_MENUBARITEM = 3

MP_CHEVRON = 5

MP_MENUDROPDOWN = 2

MP_MENUITEM = 1

MP_SEPARATOR = 6

MENUBAND

MDP_NEWAPPBUTTON = 1

MDP_SEPERATOR = 2

PAGE

PGRP_DOWN = 2

PGRP_DOWNHORZ = 4

PGRP_UP = 1

PGRP_UPHORZ = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MDS_NORMAL = 1

= 2 MDS_PRESSEI

MDS_DISABLED =

MDS_CHECKED =

MDS_HOTCHECKE

MDS_DISABLED =

MDS_CHECKED =

MDS_HOTCHECKE

DNS_NORMAL = 1

= 2 DNS_PRESSEI

DNS_DISABLED =

DNHZS_NORMAL =

DNHZS_HOT = 2

DNHZS_PRESSED

DNHZS_DISABLED

UPS_NORMAL = 1

= 2 UPS_PRESSEI

UPS_DISABLED =

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
RP_CHEVRONVERT = 5
RP_GRIPPER = 1
RP_GRIPPERVERT = 2

CHEVS_NORMAL =
CHEVS_HOT = 2
CHEVS_PRESSED

SCROLLBAR

SBP_ARROWBTN = 1
SBP_GRIPPERHORZ = 8
SBP_GRIPPERVERT = 9
SBP_LOWERTRACKHORZ = 4
SBP_LOWERTRACKVERT = 6
SBP_THUMBBTNHORZ = 2

ABS_DOWNDISAB
ABS_DOWNHOT,
ABS_DOWNNORM
ABS_DOWNPRESS
ABS_UPDISABLED
ABS_UPHOT,
ABS_UPNORMAL,
ABS_UPPRESSED,
ABS_LEFTDISABLI
ABS_LEFTHOT,
ABS_LEFTNORMA
ABS_LEFTPRESSE
ABS_RIGHTDISAB
ABS_RIGHTHOT,
ABS_RIGHTNORM
ABS_RIGHTPRESS

SCRBS_NORMAL =
SCRBS_HOT = 2
SCRBS_PRESSED
SCRBS_DISABLED
SCRBS_NORMAL =
SCRBS_HOT = 2
SCRBS_PRESSED
SCRBS_DISABLED
SCRBS_NORMAL =
SCRBS_HOT = 2
SCRBS_PRESSED

SBP_THUMBBTNVERT = 3

SBP_UPPERTRACKHORZ = 5

SBP_UPPERTRACKVERT = 7

SBP_SIZEBOX = 10

SPIN

SPNP_DOWN = 2

SPNP_DOWNHORZ = 4

SPNP_UP = 1

SPNP_UPHORZ = 3

STARTPANEL

SPP_LOGOFF = 8

SPP_LOGOFFBUTTONS = 9

SPP_MOREPROGRAMS = 2

SPP_MOREPROGRAMSARROW = 3

SPP_PLACESLIST = 6

SPP_PLACESLISTSEPARATOR = 7

SPP_PREVIEW = 11

SPP_PROGLIST = 4

SCRBS_DISABLED

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SZB_RIGHTALIGN

SZB_LEFTALIGN =

DNS_NORMAL = 1

= 2 DNS_PRESSE

DNS_DISABLED =

DNHZS_NORMAL :

DNHZS_HOT = 2

DNHZS_PRESSED

DNHZS_DISABLED

UPS_NORMAL = 1

= 2 UPS_PRESSE

UPS_DISABLED =

UPHZS_NORMAL :

UPHZS_HOT = 2

UPHZS_PRESSED

UPHZS_DISABLED

SPLS_NORMAL =

SPLS_HOT = 2

SPLS_PRESSED =

SPS_NORMAL = 1

= 2 SPS_PRESSE

STATUS

SPP_PROGLISTSEPARATOR = 5

SPP_USERPANE = 1

SPP_USERPICTURE = 10

SP_GRIPPER = 3

SP_PANE = 1

SP_GRIPPERPANE = 2

TAB

TABP_BODY = 10

TABP_PANE = 9

TABP_TABITEM = 1

TABP_TABITEMBOTHEDGE = 4

TABP_TABITEMLEFTEDGE = 2

TABP_TABITEMRIGHTEDGE = 3

TABP_TOPTABITEM = 5

TABP_TOPTABITEMBOTHEDGE = 8

TABP_TOPTABITEMLEFTEDGE = 6

TIS_NORMAL = 1

TIS_SELECTED = 2

TIS_DISABLED = 4

TIS_FOCUSED = 5

TIBES_NORMAL =

TIBES_HOT = 2

TIBES_SELECTED

TIBES_DISABLED

TIBES_FOCUSED

TILES_NORMAL =

TILES_HOT = 2

TILES_SELECTED

TILES_DISABLED

TILES_FOCUSED

TIRES_NORMAL =

TIRES_HOT = 2

TIRES_SELECTED

TIRES_DISABLED

TIRES_FOCUSED

TTIS_NORMAL = 1

TTIS_SELECTED = 2

TTIS_DISABLED =

TTIS_FOCUSED =

TTIBES_NORMAL

TTIBES_HOT = 2

TTIBES_SELECTED

TTIBES_DISABLED

TTIBES_FOCUSED

TTILES_NORMAL

TTILES_HOT = 2

TTILES_SELECTED

TTILES_DISABLED

TTILES_FOCUSED

TABP_TOPTABITEMRIGHTEDGE = 7

TTIRES_NORMAL
TTIRES_HOT = 2
TTIRES_SELECTE
TTIRES_DISABLED
TTIRES_FOCUSEC

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

TP_DROPDOWNBUTTON = 2

TP_SPLITBUTTON = 3

TP_SPLITBUTTONDROPDOWN = 4

TP_SEPARATOR = 5

TS_NORMAL = 1 T
TS_PRESSED = 3
TS_DISABLED = 4
TS_CHECKED = 5
TS_HOTCHECKED
TS_NORMAL = 1 T
TS_PRESSED = 3
TS_DISABLED = 4
TS_CHECKED = 5
TS_HOTCHECKED
TS_NORMAL = 1 T
TS_PRESSED = 3
TS_DISABLED = 4
TS_CHECKED = 5
TS_HOTCHECKED
TS_NORMAL = 1 T
TS_PRESSED = 3
TS_DISABLED = 4
TS_CHECKED = 5
TS_HOTCHECKED
TS_NORMAL = 1 T
TS_PRESSED = 3
TS_DISABLED = 4
TS_CHECKED = 5

TP_SEPARATORVERT = 6

TS_HOTCHECKED
TS_NORMAL = 1
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

TTP_STANDARD = 1

TTCS_PRESSED =
TTSS_NORMAL =

TTP_STANDARDTITLE = 2

TTSS_LINK = 2
TTSS_NORMAL =

TRACKBAR

TKP_THUMB = 3

TUS_NORMAL = 1
2 TUS_PRESSED =

TKP_THUMBBOTTOM = 4

TUS_FOCUSED =
TUS_DISABLED =

TKP_THUMBLEFT = 7

TUBS_NORMAL =
TUBS_HOT = 2

TKP_THUMBRIGHT = 8

TUBS_PRESSED =
TUBS_FOCUSED =

TKP_THUMBTOP = 5

TUBS_DISABLED =
TUVLS_NORMAL =

TUVLS_HOT = 2
TUVLS_PRESSED

TUVLS_FOCUSED
TUVLS_DISABLED

TUVRS_NORMAL =
TUVRS_HOT = 2

TUVRS_PRESSED
TUVRS_FOCUSED

TUVRS_DISABLED
TUTS_NORMAL =

TUTS_HOT = 2
TUTS_PRESSED =

TUTS_FOCUSED =

TKP_THUMBVERT = 6

TKP_TICS = 9

TKP_TICSVERT = 10

TKP_TRACK = 1

TKP_TRACKVERT = 2

TRAYNOTIFY

TNP_ANIMBACKGROUND = 2

TNP_BACKGROUND = 1

TREEVIEW

TVP_BRANCH = 3

TVP_GLYPH = 2

TVP_TREEITEM = 1

WINDOW

WP_CAPTION = 1

WP_CAPTIONSIIZINGTEMPLATE = 30

WP_CLOSEBUTTON = 18

WP_DIALOG = 29

WP_FRAMEBOTTOM = 9

WP_FRAMEBOTTOMSIIZINGTEMPLATE = 36

WP_FRAMELEFT = 7

WP_FRAMELEFTSIIZINGTEMPLATE = 32

WP_FRAMERIGHT = 8

WP_FRAMERIGHTSIIZINGTEMPLATE = 34

WP_HELPBUTTON = 23

TUTS_DISABLED =

TUVS_NORMAL =

TUVS_HOT = 2

TUVS_PRESSED =

TUVS_FOCUSED =

TUVS_DISABLED =

TSS_NORMAL = 1

TSVS_NORMAL =

TRS_NORMAL = 1

TRVS_NORMAL =

GLPS_CLOSED =

GLPS_OPENED =

TREIS_NORMAL =

TREIS_HOT = 2

TREIS_SELECTED

TREIS_DISABLED

TREIS_SELECTED

= 5

CS_ACTIVE = 1 CS

= 2 CS_DISABLED

CBS_NORMAL = 1

= 2 CBS_PUSHED

CBS_DISABLED =

FS_ACTIVE = 1 FS

= 2

FS_ACTIVE = 1 FS

= 2

FS_ACTIVE = 1 FS

= 2

HBS_NORMAL = 1

= 2 HBS_PUSHED

WP_HORIZSCROLL = 25

WP_HORIZTHUMB = 26

WP_MAX_BUTTON

WP_MAXCAPTION = 5

WP_MDICLOSEBUTTON = 20

WP_MDIHELPBUTTON = 24

WP_MDIMINBUTTON = 16

WP_MDIRESTOREBUTTON = 22

WP_MDISYSBUTTON = 14

WP_MINBUTTON = 15

WP_MINCAPTION = 3

WP_RESTOREBUTTON = 21

HBS_DISABLED =
HSS_NORMAL = 1
= 2 HSS_PUSHED =
HSS_DISABLED =
HTS_NORMAL = 1
2 HTS_PUSHED =
HTS_DISABLED =
MAXBS_NORMAL =
MAXBS_HOT = 2
MAXBS_PUSHED =
MAXBS_DISABLED =
MXCS_ACTIVE = 1
MXCS_INACTIVE =
MXCS_DISABLED =
CBS_NORMAL = 1
= 2 CBS_PUSHED =
CBS_DISABLED =
HBS_NORMAL = 1
= 2 HBS_PUSHED =
HBS_DISABLED =
MINBS_NORMAL =
MINBS_HOT = 2
MINBS_PUSHED =
MINBS_DISABLED =
RBS_NORMAL = 1
= 2 RBS_PUSHED =
RBS_DISABLED =
SBS_NORMAL = 1
= 2 SBS_PUSHED =
SBS_DISABLED =
MINBS_NORMAL =
MINBS_HOT = 2
MINBS_PUSHED =
MINBS_DISABLED =
MNCS_ACTIVE = 1
MNCS_INACTIVE =
MNCS_DISABLED =
RBS_NORMAL = 1
= 2 RBS_PUSHED =
RBS_DISABLED =
CS_ACTIVE = 1 CS

WP_SMALLCAPTION = 2	= 2 CS_DISABLED
WP_SMALLCAPTIONSIZINGTEMPLATE = 31	
WP_SMALLCLOSEBUTTON = 19	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_SMALLFRAMEBOTTOM = 12	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37	
WP_SMALLFRAMELEFT = 10	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMELEFTSIZINGTEMPLATE = 33	
WP_SMALLFRAMERIGHT = 11	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35	
WP_SMALLHELPBUTTON	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON	MAXBS_NORMAL : MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =

WP_VERTSCROLL = 27

WP_VERTTHUMB = 28

VSS_NORMAL = 1
= 2 VSS_PUSHED =
VSS_DISABLED =
VTS_NORMAL = 1
2 VTS_PUSHED =
VTS_DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Type	Description
------	-------------

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

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

- control's borders, [Appearance](#) property
- control's **header bar**, [BackColorHeader](#) property
- control's **filter bar**, [FilterBarBackColor](#) property
- **selected item** or cell, [SelBackColor](#) property
- **item**, [ItemBackColor](#) property
- **cell**, [CellBackColor](#) property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltips, and so on, [Background](#) property
- and so on.

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

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

- control's borders, [Appearance](#) property
- control's **header bar**, [BackColorHeader](#) property
- control's **filter bar**, [FilterBarBackColor](#) property
- **selected item** or cell, [SelBackColor](#) property
- **item**, [ItemBackColor](#) property
- **cell**, [CellBackColor](#) property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltips, and so on, [Background](#) property
- and so on.

Column object

The group object supports multiple columns. The Columns object contains a collection of Column objects. By default, the group adds a column. The Column object holds information about a control's column like: Alignment, Caption, Position and so on. The Column object supports the following properties and methods:

Name	Description
Alignment	Retrieves or sets the alignment of the caption into the column's header.
AllowDragging	Retrieves or sets a value indicating whether the user will be able to drag the column.
AllowSizing	Retrieves or sets a value indicating whether the user will be able to change the width of the visible columns by dragging.
AutoSearch	Specifies the kind of searching while user types characters within the columns.
AutoWidth	Computes the column's width required to fit the entire group's client area.
Caption	Retrieves or sets the text displayed in 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.
DisplayFilterButton	Specifies whether the column's header displays the filter button.
DisplayFilterDate	Specifies whether the drop down filter window displays a date selector to specify the interval dates to filter for.
DisplayFilterPattern	Specifies whether the dropdown filter bar 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.
Filter	Specifies the column's filter when filter type is exFilter, exPattern or exDate.
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.
FireFormatColumn	Retrieves or sets a value that indicates whether the control fires FormatColumn to format the caption of a cell hosted by column.
FormatColumn	Specifies the format to display the cells in the column.
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 into 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..
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 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.

[ShowFilter](#)

Shows the column's filter window.

[SortOrder](#)

Specifies the column's sort order.

[SortType](#)

Returns or sets a value that indicates the way a group 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 into the column's header.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the cells inside the column.

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 group's hierarchy lines. Use the [CellHAlignment](#) property to align a particular cell.

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.

property `Column.AllowSizing` as Boolean

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

Type	Description
Boolean	A boolean expression indicating 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 `Group` object to fit the columns to the group's client area.

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 Group object is True. Use the AutoSearch property to define a 'contains' incremental search. If the AutoSearch property is exContains, the group searches for items that contains the typed characters. The searching column is defined by the [SearchColumnIndex](#) property.

property Column.AutoWidth as Long

Computes the column's width required to fit the entire group's client area

Type	Description
Long	A long expression that indicates the width of the column to fit the entire group's client area.

Use the AutoWidth property to arrange the columns to fit the entire group's content. The AutoWidth property doesn't change the column's width. Use [Width](#) property to change the column's width at runtime.

```
Private Sub autoSize(ByVal t As EXPLORERTREELibCtl.Group)
    t.BeginUpdate
        Dim c As Column
        For Each c In t.Columns
            c.Width = c.AutoWidth
        Next
    t.EndUpdate
    t.Refresh
End Sub
```


property Column.Caption as String

Retrieves or sets the text displayed to 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. The column's caption is displayed using the following font attributes: [HeaderBold](#), [HeaderItalic](#), [HeaderUnderline](#), [HeaderStrikeout](#)

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 CellCaptionFormat 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 [CellCaptionFormat](#) property on exText, uses the same formula to display their content. For instance, you can use the CellCaptionFormat property on exHTML, for cells in the column, that need to display other things than column's formula, or you can use the CellCaptionFormat property on exComputedField, to change the formula for a particular cell.

Use the CellCaptionFormat property to change the type for a particular cell. Use the [CellCaption](#) property to specify the cell's content. For instance, if the CellCaptionFormat property is exComputedField, the Caption property indicates the formula to compute the cell's content.

The [Def](#)(exCellCaptionFormat) 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](#)(exCellCaptionFormat) property is set to exText. Call the [Refresh](#) method to force refreshing the group.

The 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).

The supported binary arithmetic operators are:

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

The supported unary boolean operators are:

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

The supported binary boolean operators are:

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

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

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

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

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

"expression ? true_part : false_part"

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

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

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

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

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

the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

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

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

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

- ***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 than using *if* and *or* expressions.

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

The supported conversion unary operators are:

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

Here's few predefined types:

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

Other known operators for numbers are:

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

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

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

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

Other known operators for strings are:

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

Other known operators for dates are:

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

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

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 ``) : ``"` displays 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.

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.

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 [Description\(exFilterBarAll\)](#) property on empty string to hide the All predefined item, in the drop down filter window. 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]]].

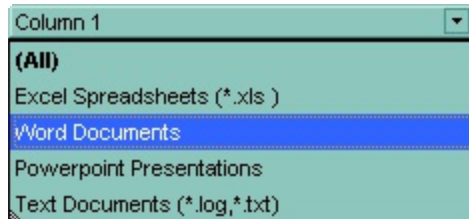
For example, you may have a list of documents and instead of listing the name of each document in the filter drop down list for the names column you may want to list the following:

- 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.

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.

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.

```
ExplorerTree1.Groups(0).Columns(0).Def(exCellHasCheckBox) = True
```

property Column.DefaultSortOrder as Boolean

Specifies whether the default sort order is ascending or descending.

Type	Description
Boolean	A boolean expression that specifies whether the default sort order is ascending or descending. True means ascending, False means descending.

By default, the DefaultSortOrder property is False. Use the [SortOnClick](#) property to specify the operation when user clicks the column's caption. Use the DefaultSortOrder to specify how the column is sorted at the first click on its header. Use the [SortOrder](#) property to sort a column.

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.

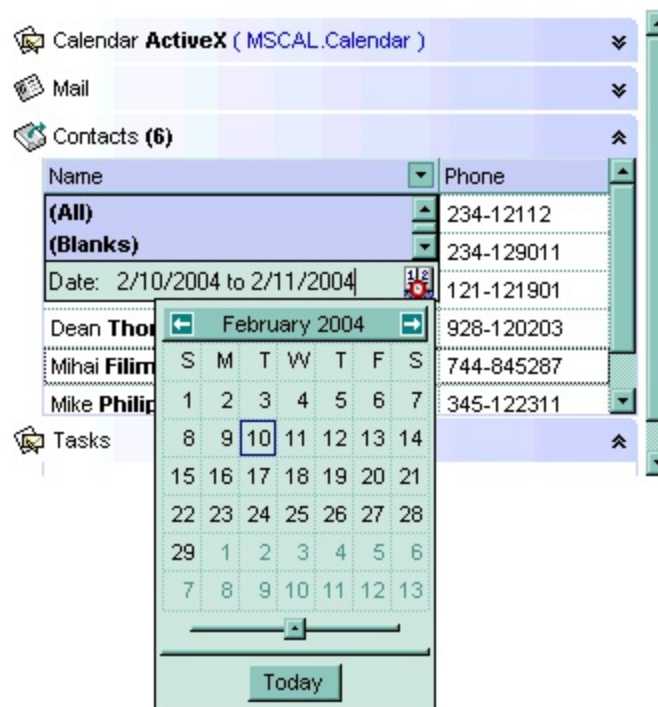
The column's filter button is displayed on the column's caption. The [DisplayFilterPattern](#) property determines whether the column's filter window includes the pattern field. Use the [FilterBarDropDownHeight](#) to specify the height of the drop down filter window. Use the [FilterBarHeight](#) property to specify the height of the filter bar header.

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. Use the [Description](#) property to customize the strings being displayed on 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 DisplayFilterPattern 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.

property Column.DisplaySortIcon as Boolean

Retrieves or sets a value indicating whether the sort icon is visible on 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, while the column is sorted.

Use the DisplaySortIcon property to hide the sort icon. Use the [SortChildren](#) property of the Items object to sort a column.

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.

If the Enabled property is False, then all cells of the column are disabled, no matter if the [CellEnabled](#) property is true. The following sample disables the first column cells randomly:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
    Group.Items.CellEnabled(Item, 0) = 4 * Rnd(4) > 2  
End Sub
```

property Column.Filter as String

Specifies the column's filter when the filter type is `exFilter` or `exPattern`.

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 [ApplyFilter](#) method is called.

- 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 group's content after changing the Filter or FilterType property. The [ClearFilter](#) method clears the Filter and the FilterType properties. Use the [CustomFilter](#) property to define you custom filters. Use the [CustomFilter](#) property to define you custom filters.

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. Use the [Description](#) property to define predefined strings in the filter bar

The following VB sample specifies that the width of the drop down filter window is double of the column's width:

```
With ExplorerTree1.Columns(0)
    .FilterBarDropDownWidth = 2
End With
```

The following VB sample specifies that the width of the drop down filter window is 150 pixels:

```
With ExplorerTree1.Columns(0)
    .FilterBarDropDownWidth = -150
End With
```

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.

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. User 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. The control fires the [FilterChange](#) event to notify whether the control applies a new filter to control's data. 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 [Description](#) property to customize the text being displayed in the drop down filter window. 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 [CustomFilter](#) property to define you custom filters.

The [ApplyFilter](#) method should be called to update the group's content after changing the Filter or FilterType property. The [ClearFilter](#) method clears the Filter and the FilterType properties.

property Column.FireFormatColumn as Boolean

Retrieves or sets a value that indicates whether the control fires FormatColumn to format the caption of a cell hosted by column.

Type	Description
Boolean	A boolean expression that indicates whether the control fires the FireFormatColumn event for the cells in the column.

By default, the FireFormatColumn property is false. The [FormatColumn](#) event is fired only if the FireFormatColumn property of the Column is True. The FormatColumn event lets the user to provide the cell's caption before it is displayed on the group's list. For instance, the FormatColumn event is very useful when the column cells contains prices (numbers), and you want to display that column formatted as currency, like \$50 instead 50.

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 [CellCaptionFormat](#) 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 [ComputedField](#) property indicates the formula of the computed column.

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>' : '') + 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):

Name	A	B	C	A+B+C
Root				
Child 1	7+	3+	1=	\$11.00
Child 2	2+	6+	12=	\$19.00
Child 3	2+	2+	4=	\$8.00
Child 4	2+	9+	4=	\$15.00

The **value** keyword in the FormatColumn property indicates the value to be formatted.

The supported binary arithmetic operators are:

- * (multiplicity operator), priority 5
- / (divide operator), priority 5
- **mod** (remainder 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** (and operator), priority 1

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

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

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

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

"expression ? true_part : false_part"

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

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

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

retrieved. The syntax for *in* operator is

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

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

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

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

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

- ***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 than using *iif* and *or* expressions.

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

The supported conversion unary operators are:

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

Here's few predefined types:

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

- 20 - long on 64 bits
- 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format " displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

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

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- *Grouping* - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using.

The valid values are 0, 1, 2, 3 and 4 with the following meanings:

- 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
- 1 - Negative sign, number; for example, -1.1
- 2 - Negative sign, space, number; for example, - 1.1
- 3 - Number, negative sign; for example, 1.1-
- 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

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

Other known operators for dates are:

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

"MM/DD/YYYY HH:MM:SS" format.

- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

*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. You can use variables to avoid computing several times the same thing.*

property Column.HeaderAlignment as AlignmentEnum

Specifies the alignment of the column's caption.

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

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.

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.

For instance, the following samples shows how to bold the entire column, by handling the [AddItem](#) event:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
    Group.Items.CellBold(Item, 0) = True  
End Sub
```

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 image in the column's header.

Use the HeaderImage property to assign an icon to the column's header. Use the [HeaderImageAlignment](#) property to align the column's icon inside the column's header.

property Column.HeaderImageAlignment as AlignmentEnum

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

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the image 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.

property Column.HeaderItalic as Boolean

Retrieves or sets the Italic property of the Font object that it is used to paint the column's caption.

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.

For instance, the following sample shows how to bold the entire column, by handling the AddItem event:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
    Group.Items.CellBold(Item, 0) = True  
End Sub
```

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**.

For instance, the following sample shows how to bold the entire column, by handling the `AddItem` event:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal  
Item As EXPLOREERTREELibCtl.HITEM)  
    Group.Items.CellBold(Item, 0) = True  
End Sub
```

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.

For instance, the following samples show how to bold the entire column, by handling the AddItem event:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal  
Item As EXPLOREERTREELibCtl.HITEM)  
    Group.Items.CellBold(Item, 0) = True  
End Sub
```

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 group uses it when displays the column's header. Use the [HeaderHeight](#) property to change the height of the group's header bar. The list of built-in HTML tags supported are [here](#).

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 represents the index of an object in a collection.

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 group updates all indexes.

property Column.Key as String

Retrieves or sets 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.

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. Use the [PictureLevelHeader](#) property to assign a picture on the control's header. The [BackColorHeader](#) property specifies the background color for column's captions.



property Column.MaxWidthAutoSize as Long

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

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

Use the MaxWidthAutoSize property to set the maximum column's width while the [WidthAutoSize](#) property is True. If the MaxWidthAutoSize property is less than zero, there is no maximum value for the column's width. By default, the MaxWidthAutoSize property is -1.

property Column.MinWidthAutoSize as Long

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

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

Use the MinWidthAutoSize property to set the minimum column's width while the [WidthAutoSize](#) property is True.

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 the 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 [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. Use the [CheckImage](#) property to define the icons for each state.

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 group's client area.

method Column.ShowFilter ([Options as Variant])

Shows the column's filter window.

Type

Description

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

Options as Variant

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 automatically computed based on the [FilterBarDropDownWidth](#) property and [FilterBarDropDownHeight](#) property. Use the [ColumnFromPoint](#) property to get the index of the column from the point.

The screenshot shows a data grid with columns: Name, A, B, C, and A+B+C. The 'Name' column is selected, and a filter window is displayed over it. The filter window lists predefined filters: (All), (Blanks), and (NonBlanks). Below these are the grid's rows: Root, Child 1, and Child 2. The 'Filter For:' field is at the bottom of the window. The grid data is as follows:

Name	A	B	C	A+B+C
Root				
Child 1	7	3	1	11
Child 2	2	6	12	19

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.

The group automatically sorts a column when the user clicks the column's header. If the [SortOnClick](#) property is False the group disables sorting the items when 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::

```
Group.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 group. SortChildren will not recurse 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 group displays a list, all items are considered being root items).

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

property Column.SortType as SortTypeEnum

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

Type	Description
SortTypeEnum	A SortTypeEnum expression that indicates the way a group sorts the values for a column.

The SortType property specifies the way how a column is sorted. By default, the column's SortType is String. Use the SortType property to specifies how the group will sort the column. Use the [SortChildren](#) property of Items to do a sort based on a column

property Column.ToolTip as String

Specifies 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 empty. Use the ToolTip property to assign a tooltip to a column. The column's tooltip shows up when the cursor is over the header of the column.

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 [Caption](#) property to change the column's caption. Use the [Position](#) property to specify the column's position.

property Column.Width as Long

Retrieves or sets the column's width.

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

The Width property specifies the column's width in pixels. Use the [Visible](#) property to hide a column.

To change the column width for all columns, you can use the following sample , by handling the AddColumn event:

```
Private Sub ExplorerTree1_AddColumn(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal  
Column As EXPLOREERTREELibCtl.IColumn)  
    Column.Width = 196  
End Sub
```

property Column.WidthAutoSize 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.

If the `WidthAutoSize` property is `True`, the column's width is resized after user expands, or collapse the items. Also, the column's width is refreshed if the user adds new items to the group. If the `WidthAutoSize` property is `True`, the column's width is not larger than [MaxWidthAutoSize](#) value, and it is not less than [MinWidthAutoSize](#) value. 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 group will automatically add a horizontal scroll bar when required.

Columns object

The Group object supports multiple columns. The Columns object contains a collection of Column objects. Use the [Columns](#) property of the control to access the group columns. By default, the group's columns collection contains a column.

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.
Remove	Removes a specific member from the Columns collection.

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 indicates the caption for the column being added

Return	Description
Variant	A Column object that indicates the newly added column.

The AddColumn event is fired when a new columns is added to Columns collection. Use the [Caption](#) property to change the column's caption.

The following sample shows how to add columns to your group based on a recordset:

```
With ExplorerTree1
  With .Groups.Add("Group 1")
    Set rs = CreateObject("ADODB.Recordset")
    rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
    .BeginUpdate
    .ColumnAutoSize = False
    ' Add the columns
    With .Columns
      For Each f In rs.Fields
        .Add f.Name
      Next
    End With
    ' Adds items
    .PutItems rs.getRows()
  .EndUpdate
End With
End With
```

method Columns.Clear ()

Removes all objects in a collection.

Type	Description
------	-------------

Use the [Remove](#) method when you need to remove only a column. Use the Clear method when you need to clear the entire columns collection. Also, the Clear method removes all items. Use the [RemoveAllItems](#) method to remove all items in the group.

property Columns.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	Counts the Column object into the collection.

You can use the following samples to enumerate the group columns:

```
Dim c As EXPLOREERTREELibCtl.Column
With ExplorerTree1.Groups(0)
  For Each c In .Columns
    Debug.Print c.Caption
  Next
End With
```

```
Dim i As Long
With ExplorerTree1.Groups(0).Columns
  For i = 0 To .Count - 1
    Debug.Print .Item(i).Caption
  Next
End With
```

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 key or the column's caption.
Column	A column object being returned.

Use the Item property to access to a specific column. The Item property is the default property of the Columns object so the following statements are equivalents:

```
ExplorerTree1.Groups(0).Columns.Item ("Freight")
```

```
ExplorerTree1.Groups(0).Columns("Freight")
```

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, or a string expression that indicates the column's caption or the column's key.

The Remove method removes a specific column in the Columns collection. Use [Clear](#) method to remove all Column objects. The [RemoveColumn](#) event is fired when a column is about to be removed.

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 strikethrough.
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 Group1.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_group.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 AxGroup1.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):

```
EXPLOREERTREELib.ConditionalFormat cf =  
axGroup1.ConditionalFormats.Add("%1+%2<%0",null);  
cf.Bold = true;  
cf.ApplyTo = (EXPLOREERTREELib.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.Group1.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 Group1.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_group.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 AxGroup1.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):

```
EXPLORERTREELib.ConditionalFormat cf =  
axGroup1.ConditionalFormats.Add("%1+%2<%0",null);  
cf.Bold = true;
```

```
cf.ApplyTo = (EXPLORERTREELib.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.Group1.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 supported binary arithmetic operators are:

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

The supported unary boolean operators are:

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

The supported binary boolean operators are:

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

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

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

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

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

"expression ? true_part : false_part"

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

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

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

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

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

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

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

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

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

- **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 than using iif and or expressions.

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

The supported conversion unary operators are:

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

specifies the cells that contains string values.

Here's few predefined types:

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

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format " displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

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

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

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startswith** (binary operator) specifies whether a string starts with specified string

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

Other known operators for dates are:

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

*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 ``) : ``"** displays 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.

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 strikethrough.
- [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.

Samples:

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 fourth 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 fourth columns is less than fifth column minus 5.

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

```
Group1.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_group.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:

```
AxGroup1.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:

```
axGroup1.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.Group1.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 Group1.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 Group1.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_group.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 AxGroup1.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):

```
EXPLOREERTREELib.ConditionalFormat cf =  
axGroup1.ConditionalFormats.Add("%1+%2<%0",null);  
cf.Italic = true;
```

```
cf.ApplyTo = (EXPLORERTREELib.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.Group1.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 Group1.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_group.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 AxGroup1.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):

```
EXPLOREERTREELib.ConditionalFormat cf =  
axGroup1.ConditionalFormats.Add("%1+%2<%0",null);  
cf.Bold = true;  
cf.ApplyTo = (EXPLOREERTREELib.FormatApplyToEnum)1;
```

The following VFP sample applies strikethrough 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.Group1.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 Group1.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_group.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 AxGroup1.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):

```
EXPLORERTREELib.ConditionalFormat cf =  
axGroup1.ConditionalFormats.Add("%1+%2<%0",null);  
cf.Underline = true;
```

```
cf.ApplyTo = (EXPLORERTREELib.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.Group1.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 strikethrough.
- [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:

```
Group1.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 Group1.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_group.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_group.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:

```
AxGroup1.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 AxGroup1.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:

```
axGroup1.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):

```
EXPLOREERTREELib.ConditionalFormat cf =  
axGroup1.ConditionalFormats.Add("%1+%2<%0",null);  
cf.Bold = true;  
cf.ApplyTo = (EXPLOREERTREELib.FormatApplyToEnum)1;
```

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

```
thisform.Group1.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.Group1.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 Group1.ConditionalFormats
    Debug.Print c.Expression
Next
```

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

```
Dim i As Integer
With Group1.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_group.GetConditionalFormats().GetCount(); i++ )
{
    CConditionalFormat cf = m_group.GetConditionalFormats().GetItem( COleVariant( i ) );
    OutputDebugString( cf.GetExpression() );
}
```

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

```
Dim c As EXPLORERTREELib.ConditionalFormat
For Each c In AxGroup1.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 AxGroup1.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 (EXPLOREERTREELib.ConditionalFormat c in axGroup1.ConditionalFormats)
  System.Diagnostics.Debug.Write(c.Expression);
```

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

```
for (int i = 0; i < axGroup1.ConditionalFormats.Count; i++)
  System.Diagnostics.Debug.Write(axGroup1.ConditionalFormats[i].Expression);
```

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

```
with thisform.Group1.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 Group1.ConditionalFormats
    Debug.Print c.Expression
Next
```

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

```
Dim i As Integer
With Group1.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_group.GetConditionalFormats().GetCount(); i++ )
{
    CConditionalFormat cf = m_group.GetConditionalFormats().GetItem( COleVariant( i ) );
    OutputDebugString( cf.GetExpression() );
}
```

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

```
Dim c As EXPLORERTREELib.ConditionalFormat
```



```
For Each c In AxGroup1.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 AxGroup1.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 (EXPLOREERTREELib.ConditionalFormat c in axGroup1.ConditionalFormats)
    System.Diagnostics.Debug.Write(c.Expression);
```

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

```
for (int i = 0; i < axGroup1.ConditionalFormats.Count; i++)
    System.Diagnostics.Debug.Write(axGroup1.ConditionalFormats[i].Expression);
```

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

```
with thisform.Group1.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.

ExDataObject object

The [OleDragDrop](#) event notifies your application that the user drags some data on the control. Defines the object that contains OLE drag and drop information. The ExDataObject object supports the following method and properties:

Name	Description
Clear	Deletes the contents of the ExDataObject object.
Files	Returns an ExDataObjectFiles collection, which in turn contains a list of all filenames used by an ExDataObject object.
GetData	Returns data from an ExDataObject object in the form of a variant.
GetFormat	Returns a value indicating whether an item in the ExDataObject object matches a specified format.
SetData	Inserts data into an ExDataObject object using the specified data format.

method `ExDataObject.Clear ()`

Deletes the contents of the `DataObject` object.

Type	Description
------	-------------

The `Clear` method can be called only for drag sources. The [OleDragDrop](#) event notifies your application that the user drags some data on the control.

property `ExDataObject.Files` as `ExDataObjectFiles`

Returns a `DataObjectFiles` collection, which in turn contains a list of all filenames used by a `DataObject` object.

Type	Description
ExDataObjectFiles	An <code>ExDataObjectFiles</code> object that contains a list of filenames used in OLE drag and drop operations.

The `Files` property is valid only if the format of the clipboard data is `exCFFiles`. The [OleDragDrop](#) event notifies your application that the user drags some data on the control.

method ExDataObject.GetData (Format as Integer)

Returns data from a DataObject object in the form of a variant.

Type	Description
Format as Integer	An exClipboardFormatEnum expression that defines the data's format
Return	Description
Variant	A Variant value that contains the ExDataObject's data in the given format

Use GetData property to retrieve the clipboard's data that has been dragged to the control. It's possible for the GetData and [SetData](#) methods to use data formats other than [exClipboardFormatEnum](#) , including user-defined formats registered with Windows via the RegisterClipboardFormat() API function. The GetData method always returns data in a byte array when it is in a format that it is not recognized. Use the [Files](#) property to retrieves the filenames if the format of data is exCFFiles

method `ExDataObject.GetFormat` (Format as Integer)

Returns a value indicating whether the `ExDataObject`'s data is of specified format.

Type	Description
Format as Integer	A constant or value that specifies a clipboard data format like described in exClipboardFormatEnum enum.
Return	Description
Boolean	A boolean value that indicates whether the <code>ExDataObject</code> 's data is of specified format.

Use the `GetFormat` property to verify if the `ExDataObject`'s data is of a specified clipboard format. The `GetFormat` property retrieves `True`, if the `ExDataObject`'s data format matches the given data format.

method `ExDataObject.SetData ([Value as Variant], [Format as Variant])`

Inserts data into a `ExDataObject` object using the specified data format.

Type	Description
Value as Variant	A data that is going to be inserted to <code>ExDataObject</code> object.
Format as Variant	A constant or value that specifies the data format, as described in exClipboardFormatEnum enum

Use `SetData` property to insert data for OLE drag and drop operations. Use the [Files](#) property if you are going to add new files to the clipboard data. The [OleDragDrop](#) event notifies your application that the user drags some data on the control.

ExDataObjectFiles object

The ExDataObjectFiles contains a collection of filenames. The ExDataObjectFiles object is used in OLE Drag and drop events. In order to get the list of files used in drag and drop operations you have to use the [Files](#) property. The [OleDragDrop](#) event notifies your application that the user drags some data on the control. The ExDataObjectFiles object supports the following properties and methods:

Name	Description
Add	Adds a filename to the Files collection
Clear	Removes all file names in the collection.
Count	Returns the number of file names in the collection.
Item	Returns an specific file name.
Remove	Removes an specific file name.

method `ExDataObjectFiles.Add (FileName as String)`

Adds a filename to the Files collection

Type	Description
FileName as String	A string expression that indicates a filename.

Use Add method to add your files to ExDataObject object. The [OleStartDrag](#) event notifies your application that the user starts dragging items.

method `ExDataObjectFiles.Clear ()`

Removes all file names in the collection.

Type	Description
------	-------------

Use the `Clear` method to remove all filenames from the collection.

property `ExDataObjectFiles.Count` as `Long`

Returns the number of file names in the collection.

Type	Description
Long	A long value that indicates the count of elements into collection.

You can use "for each" statements if you are going to enumerate the elements into `ExDataObjectFiles` collection.

property `ExDataObjectFiles.Item (Index as Long) as String`

Returns a specific file name given its index.

Type	Description
Index as Long	A long expression that indicates the filename's index.
String	A string value that indicates the filename.

method `ExDataObjectFiles.Remove (Index as Long)`

Removes a specific file name given its index into collection.

Type	Description
Index as Long	A long expression that indicates the index of filename into collection.

Use [Clear](#) method to remove all filenames,.

ExplorerTree 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: {1036744E-4103-4987-BA7A-BB6C35BD5852}. The object's program identifier is: "Exontrol.ExplorerTree". The /COM object module is: "ExplorerTree.dll"

Add structured navigation functionality to your applications. The ExplorerTree component adds navigation functionality to your applications, it brings simple information structuring and easy application navigation. It contains a WYSWYG designer, which is available in all environments such as .NET, VFP or else. It simplifies the organization of information in your applications. Hierarchical structure of Groups and Items allows perfect structuring of information. Create Outlook style bar and tree navigation interfaces. The ExplorerTree supports the following properties and methods:

Name	Description
AllowResizeShortcutBar	Specifies whether the user can resize the shortcutbar, to allow multiple shortcuts to be visible.
AllowTooltip	Specifies whether the control displays a tooltip when the item's caption is too long.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Specifies the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoScrollBar	Specifies whether the control adds the vertical scroll bar if required.
BackColor	Retrieves or sets a value that indicates the control's background color.
BackColorGroup	Retrieves or sets a value that indicates the group's background color.
BackColorGroup2	Specifies the color at the ending boundary line of the gradient group's caption.
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.
BorderGroupHeight	Specifies the height of the border between groups.
BorderHeight	Specifies the border's height.
BorderWidth	Specifies the border's width.
DelayScroll	Specifies the delay used when user selects a group.

DisplayExpandIcon	Specifies whether the control displays the expand icon on the group's caption.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EnsureVisible	Ensures the given group/item is in the visible client area.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
ExpandIcon	Retrieves or sets the index of the icon used to paint the expand button.
ExpandOnClick	Expands the group when its caption is clicked.
ExpandShortcutCount	Retrieves or sets a value that indicates the number of shortcuts being expanded.
ExpandShortcutImage	Retrieves or sets a value that indicates the index of the image being displayed to expand the shortcuts.
FocusGroup	Retrieves the group that has the focus.
Font	Retrieves or sets the control's font.
ForeColor	Retrieves or sets a value that indicates the control's foreground color.
ForeColorGroup	Retrieves or sets a value that indicates the group's foreground color.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
GroupAppearance	Specifies the group's appearance.
GroupFromPoint	Gets the group from point.
GroupHeight	Specifies the group's height.
GroupListFromPoint	Gets the group's list from point.
Groups	Retrieves the control's groups collection.
HandCursor	Specifies whether the control uses the hand cursor when the mouse is over the group.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the handle of the control's window.
Images	Sets the control's handle image list.

ImageSize	Retrieves or sets the size of icons the control displays.
OLEDrag	Causes a component to initiate an OLE drag/drop operation.
OLEDropMode	Returns or sets how a target component handles drop operations
Picture	Retrieves or sets a graphic to be displayed in the control's background.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background.
Refresh	Refreshes the control.
Replacelcon	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.
SelectShortcut	Selects and displays the specified shortcut.
ShortcutBarBackColor	Retrieves or sets the shortcut bar's background color.
ShortcutBarHeight	Selects and displays the specified shortcut.
ShortcutBarSelBackColor	Retrieves or sets the background color for the selected icon in the shortcut bar.
ShortcutBarSelCaptionBackColor	Retrieves or sets the background color for selected shortcut when its entire caption is displayed.

ShortcutFromPoint	Gets the shortcut from the cursor.
ShortcutPicture	Specifies a custom-size picture assigned to a shortcut.
ShortcutPictureHeight	Specifies the height in pixels of the custom size picture being displayed in the shortcut bar.
ShortcutPictureWidth	Specifies the width in pixels of the custom size picture being displayed in the shortcut bar.
ShortcutResizeBackColor	Retrieves or sets the background color for the shortcut's resize bar.
ShowFocusRect	Specifies a value that indicates whether the control draws the focused item.
ShowImageList	Retrieves or sets a value that indicates whether the image list window is visible or hidden.
ShowShortcutBar	Retrieves or sets a value that indicates whether the image shortcut bar is visible or hidden.
ShowToolTip	Shows programmatically a tooltip at given position.
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.
ToolTipMargin	Defines the size of the control's tooltip margins.
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.
UseVisualStyle	Specifies whether the control uses the current visual theme to display certain UI parts.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.

property ExplorerTree.AllowResizeShortcutBar as Boolean

Specifies whether the user can resize the shortcutbar, to allow multiple shortcuts to be visible.

Type	Description
Boolean	A Boolean expression that indicates whether the user can resize the shortcut bar.

By default, the AllowResizeShortcutBar property is True. Use the AllowResizeShortcutBar property to hide the resize bar of the control's shortcut bar. Use the [ShowShortcutBar](#) property to show or hide the control's shortcut bar. The [ShortcutResizeBackColor](#) property changes the visual appearance of the resizing bar of the shortcut bar. Use the [ExpandShortcutCount](#) property to specify the number of shortcuts that display their full caption, else the first icon in the caption is displayed or the assigned picture is displayed. The [ExpandShortcut](#) event notifies your application when the user resizes the control's shortcut bar.

property ExplorerTree.AllowTooltip as Boolean

Specifies whether the control displays a tooltip when the item's caption is too long.

Type	Description
Boolean	A boolean expression that indicates whether the control displays a tooltip when the item's caption is too long

By default, the AllowTooltip property is True. Use the AllowTooltip property to disable tooltips in the control. Use the [ToolTipWidth](#) property to specify the width of the tooltip window.

property ExplorerTree.AnchorFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as String

Retrieves the identifier of the anchor from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
String	A String expression that specifies the identifier (id) of the anchor element from the point, or empty string if there is no anchor element at the cursor.

Use the AnchorFromPoint property to determine the identifier of the anchor from the point. Use the [<a id;options>](#) anchor elements to add hyperlinks to cell's caption. The control fires the [AnchorClick](#) event when the user clicks an anchor element. Use the [ShowToolTip](#) method to show the specified tooltip at given or cursor coordinates. The [MouseMove](#) event is generated continually as the mouse pointer moves across the control.

The following VB sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ExplorerTree1
        .ShowToolTip .AnchorFromPoint(-1, -1)
    End With
End Sub
```

The following VB.NET sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub AxExplorerTree1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXPLORERTREELib._IExplorerTreeEvents_MouseMoveEvent) Handles AxExplorerTree1.MouseMoveEvent
    With AxExplorerTree1
        .ShowToolTip(.get_AnchorFromPoint(-1, -1))
    End With
End Sub
```

The following C# sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
private void axExplorerTree1_MouseMoveEvent(object sender,
AxEXEXPLORERTREELib._IExplorerTreeEvents_MouseMoveEvent e)
{
    axExplorerTree1.ShowToolTip(axExplorerTree1.get_AnchorFromPoint(-1, -1));
}
```

The following C++ sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
void OnMouseMoveExplorerTree1(short Button, short Shift, long X, long Y)
{
    COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
    m_explorerTree.ShowToolTip( m_explorerTree.GetAnchorFromPoint( -1, -1 ), vtEmpty,
vtEmpty, vtEmpty );
}
```

The following VFP sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform
    With .ExplorerTree1
        .ShowToolTip(.AnchorFromPoint(-1, -1))
    EndWith
endwith
```

property ExplorerTree.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 groups, list/hierarchy, scrollbars are always shown in the control's client area. The skin may contain transparent objects, and so you can define round corners. The [frame.ebn](#) file contains such of objects. Use the [eXButton's Skin builder](#) to view or change this file***

Use the Appearance property to specify the control's border. Use the [Add](#) method to add new skins to the control. Use the [BackColor](#) property to specify the control's background color. Use the [Background\(exToolTipAppearance\)](#) property indicates the visual appearance of the borders of the tooltips.



The following VB sample changes the visual aspect of the borders of the control (please check the above picture for round corners):

With ExplorerTree1

```
.BeginUpdate  
  .VisualAppearance.Add &H16, "c:\temp\frame.ebn"  
  .Appearance = &H16000000  
  .BackColor = RGB(250, 250, 250)  
.EndUpdate
```

End With

The following VB.NET sample changes the visual aspect of the borders of the control:

With AxExplorerTree1

```
.BeginUpdate()  
.VisualAppearance.Add(&H16, "c:\temp\frame.ebn")  
.Appearance = &H16000000  
.BackColor = Color.FromArgb(250, 250, 250)  
.EndUpdate()
```

End With

The following C# sample changes the visual aspect of the borders of the control:

```
axExplorerTree1.BeginUpdate();  
axExplorerTree1.VisualAppearance.Add(0x16, "c:\\temp\\frame.ebn");  
axExplorerTree1.Appearance = (EXEXPLORERTREELib.AppearanceEnum)0x16000000;  
axExplorerTree1.BackColor = Color.FromArgb(250, 250, 250);  
axExplorerTree1.EndUpdate();
```

The following C++ sample changes the visual aspect of the borders of the control:

```
m_explorerTree.BeginUpdate();  
m_explorerTree.GetVisualAppearance().Add( 0x16, COleVariant( "c:\\temp\\frame.ebn" ) );  
m_explorerTree.SetAppearance( 0x16000000 );  
m_explorerTree.SetBackColor( RGB(250,250,250) );  
m_explorerTree.EndUpdate();
```

The following VFP sample changes the visual aspect of the borders of the control:

with thisform.ExplorerTree1

```
.BeginUpdate  
  .VisualAppearance.Add(0x16, "c:\temp\frame.ebn")
```



```
.Appearance = 0x16000000
```

```
.BackColor = RGB(250, 250, 250)
```

```
.EndUpdate
```

```
endwith
```

method ExplorerTree.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 ExplorerTree1_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 ExplorerTree.AutoScrollBar as Boolean

Specifies whether the control adds the vertical scroll bar if required.

Type	Description
Boolean	A Boolean expression that indicates whether the control displays the vertical scroll bar when it is required.

By default, the AutoScrollBar property is True, that means that the vertical scroll bar for the control is shown when it is required. Use the AutoScrollBar property on False, to hide the control's vertical scroll bar. The control's vertical scroll bar may scroll the groups. The scrollbars in the groups scrolls items inside the groups. Use the [ScrollBars](#) property of the [Group](#) object to define the scrollbars being displayed in the groups. If the AutoScrollBar property is True, the control shows the vertical scroll bar only if required. Use the [ScrollWidth](#) property to specify the width of the vertical scroll bar. Use the [ScrollThumbSize](#) property to define a fixed size for the scrollbar's thumb. Use the [Background](#) property to change the visual aspect for the scrollbars.

property ExplorerTree.BackColor as Color

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

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

Use the BackColor property to specify the control's background color. Use the [BackColorGroup](#) property to specify the default background color used to paint the groups captions. Use the [BackColor](#) property to specify the background color group's caption, Use the [BackColorList](#) property to specify the background color of the group's list. Use the [ItemBackColor](#) property to specify the item's background color. Use the [Picture](#) property to specify the control's picture displayed on its background. Use the group's [Picture](#) property to assign a picture the group's list.

property ExplorerTree.BackColorGroup as Color

Retrieves or sets a value that indicates the default group's background color.

Type	Description
Color	A color expression that indicates the group'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.

Use the BackColorGroup property to specify the background color for all groups. Use the [BackColorGroup2](#) property to specify the second background color when drawing the caption of group like a gradient. The BackColorGroup property changes only the background color of group captions. Use the [BackColor](#), [BackColorList](#) property to change the background color for a specific group. Use the group's [Picture](#) property to assign a picture the group's list.

property ExplorerTree.BackColorGroup2 as Color

Specifies the color at the ending boundary line of the gradient group's caption.

Type	Description
Color	A color expression that specifies the color at the ending boundary line of the gradient group's caption.

Use the [BackColorGroup](#) and BackColorGroup2 properties to display the caption of the group using a gradient color.

property ExplorerTree.Background(Part as BackgroundPartEnum) as Color

Returns or sets a value that indicates the background color for parts in the control.

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

The Background property specifies a background color or a visual appearance for specific parts in the control. If the Background property is 0, the control draws the part as default. Use the [Add](#) method to add new skins to the control. Use the [Remove](#) method to remove a specific skin from the control. Use the [Clear](#) method to remove all skins in the control. Use the [BeginUpdate](#) and [EndUpdate](#) methods to maintain performance while init the control.

method ExplorerTree.BeginUpdate ()

Maintains performance when groups are added to the control one at a time.

Type

Description

Use the `BeginUpdate` and [EndUpdate](#) methods to avoid drawing the control while adding multiple groups. Use the group's [BeginUpdate](#) and [EndUpdate](#) method to maintain performance while adding new items to the group's list.

For instance, the following sample shows how to use the `BeginUpdate` and `EndUpdate` methods:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
With ExplorerTree1
    .BeginUpdate
        With .Groups.Add("Group 1")
            .BeginUpdate
                .ColumnAutoResize = False
                .HeaderVisible = True
                With .Columns
                    .Clear
                    For Each f In rs.Fields
                        .Add f.Name
                    Next
                End With
                .PutItems rs.GetRows()
            .EndUpdate
        End With
        With .Groups.Add("Group 2")
            .BeginUpdate
                .PutItems Array("Item 1", "Item 2", "Item 3")
            .EndUpdate
        End With
    .EndUpdate
End With
```


property ExplorerTree.BorderGroupHeight as Long

Specifies the height of the border between groups.

Type	Description
Long	A long expression that specifies the height of the border between groups.

By default the BorderGroupHeight property is 2 pixels. The group's list client area is computed based on the [BorderWidth](#), [BorderHeight](#), [IndentGroupLeft](#), [IndentGroupRight](#) and BorderGroupHeight properties.

property ExplorerTree.BorderHeight as Long

Specifies the border's height.

Type	Description
Long	A long expression that indicates the border's height.

By default, the BorderHeight property is 2 pixels. The group's list client area is computed based on the [BorderWidth](#), BorderHeight, [IndentGroupLeft](#), [IndentGroupRight](#) and [BorderGroupHeight](#) properties.

property ExplorerTree.BorderWidth as Long

Specifies the border's width.

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

By default, the BorderWidth property is 2 pixels. The group's list client area is computed based on the BorderWidth, [BorderHeight](#), [IndentGroupLeft](#), [IndentGroupRight](#) and [BorderGroupHeight](#) properties.

property ExplorerTree.DelayScroll as Long

Specifies the delay used when user selects a group.

Type	Description
Long	A long expression that indicates the delay used by the control when a new group is selected.

By default, the DelayScroll property is 50. If the DelayScroll property is not zero, the groups shows by animation when a group is being expanded or collapsed. If the DelayScroll property is 0, the group's view is shown directly whit no animation.

property ExplorerTree.DisplayExpandIcon as Boolean

Specifies whether the control displays the expand icon on the group's caption.

Type	Description
Boolean	A boolean expression that specifies whether the control displays the expand icon on the group's caption.

By default, the DisplayExpandIcon property is True. Use the DisplayExpandIcon property to hide the icon in the group caption that indicates whether a group is expanded or collapsed. Use the [ExpandIcon](#) property to specify the icons used for expanded/collapsed groups.

property ExplorerTree.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.

method ExplorerTree.EndUpdate ()

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

Type

Description

Use the [BeginUpdate](#) and EndUpdate methods when you do multiple changes. Use the group's [BeginUpdate](#) and [EndUpdate](#) method to maintain performance while adding new items to the group's list.

For instance, the following sample shows how to use the BeginUpdate and EndUpdate methods:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
With ExplorerTree1
    .BeginUpdate
        With .Groups.Add("Group 1")
            .BeginUpdate
                .ColumnAutoResize = False
                .HeaderVisible = True
                With .Columns
                    .Clear
                    For Each f In rs.Fields
                        .Add f.Name
                    Next
                End With
                .PutItems rs.GetRows()
            .EndUpdate
        End With
        With .Groups.Add("Group 2")
            .BeginUpdate
                .PutItems Array("Item 1", "Item 2", "Item 3")
            .EndUpdate
        End With
    .EndUpdate
End With
```

method ExplorerTree.EnsureVisible ([Group as Variant], [Item as Variant])

Ensures the given group/item is in the visible client area.

Type	Description
Group as Variant	A Long expression that specifies the index of the Group.
Item as Variant	A Long expression that specifies the index of the Item within the Group.

The EnsureVisible method ensures that giving group/item fits the control's client area.

property ExplorerTree.EventParam(Parameter as Long) as Variant

Retrieves or sets a value that indicates the current's event parameter.

Type	Description
Parameter as Long	A long expression that indicates the index of the parameter being requested ie 0 means the first parameter, 1 means the second, and so on. If -1 is used the EventParam property retrieves the number of parameters. Accessing an not-existing parameter produces an OLE error, such as invalid pointer (E_POINTER)
Variant	A VARIANT expression that specifies the parameter's value.

The EventParam method is provided to allow changing the event's parameters passed by reference, even if your environment does not support changing it (uniPaas 1.5 (formerly known as eDeveloper), DBase, and so on). For instance, Unipaas event-handling logic cannot update ActiveX control variables by updating the received arguments. The EventParam(0) retrieves the value of the first parameter of the event, while the EventParam(1) = 0, changes the value of the second parameter to 0 (the operation is successfully, only if the parameter is passed by reference). The EventParam(-1) retrieves the number of the parameters of the current event.

Let's take the event "event KeyDown (**KeyCode** as Integer, ByVal Shift as Integer)", where the KeyCode parameter is passed by reference. For instance, put the KeyCode parameter on 0, and the arrow keys are disabled while the control has the focus.

In most languages you will type something like:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
    KeyCode = 0
End Sub
```

In case your environment does not support events with parameters by reference, you can use a code like follows:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
    Control1.EventParam(0) = 0
End Sub
```

In other words, the EventParam property provides the parameters of the current event for reading or writing access, even if your environment does not allow changing parameters by

reference.

Calling the EventParam property outside of an event produces an OLE error, such as pointer invalid, as its scope was designed to be used only during events.

method ExplorerTree.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 number of groups in the control:

```
Debug.Print ExplorerTree1.ExecuteTemplate("Groups.Count")
```

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 ExplorerTree.ExpandIcon(Expanded as Boolean) as Long

Retrieves or sets the index of the icon used to paint the expand button.

Type	Description
Expanded as Boolean	A boolean expression that indicates the state of the icon being changed. True means expanded, false means collapsed.
Long	A long expression that indicates the index of icon used.

Use the ExpandIcon property to change the icon used to paint an expanded/collapsed groups. Use the [DisplayExpandIcon](#) property to hide the expanded/collapsed icon.

property ExplorerTree.ExpandOnClick as Boolean

Expands the group when its caption is clicked.

Type	Description
Boolean	A boolean expression that indicates whether the control expands or collapses the group when the group's caption is clicked.

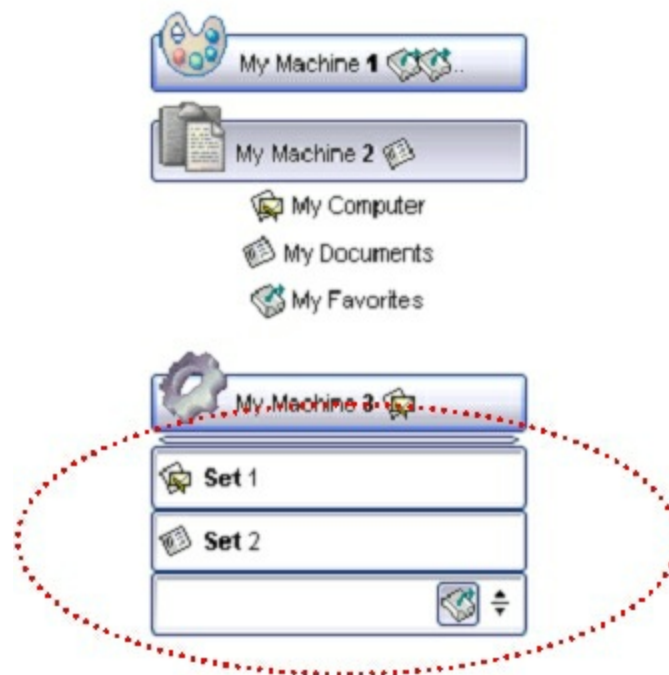
Use the ExpandOnClick property to specify whether the control expands or collapses when the user clicks the group's caption. Use the [ExpandIcon](#) property to assign new expanded/collapsed icons for the groups.

property ExplorerTree.ExpandShortcutCount as Long

Retrieves or sets a value that indicates the number of shortcuts being expanded.

Type	Description
Long	A long expression that indicates the number of shortcuts that display their full caption.

By default, the `ExpandShortcutCount` property is 0. The `ExpandShortcutCount` property is changed when the user resizes the shortcut bar. Use the [ShowShortcutBar](#) property to show or hide the control's shortcut bar. Use the [AllowResizeShortcutBar](#) property to enable or disable resizing the shortcut bar. Use the [ExpandShortcutImage](#) property to hide the expand button in the control's shortcut bar, or to change the icon of the expand button in the shortcut bar. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same `Shortcut` property are displayed in the same shortcut. The [ExpandShortcut](#) event notifies your application when the user resizes the control's shortcut bar. The control fires the [SelectShortcut](#) event when the user selects a shortcut, so groups that belongs to the shortcut are displayed.



property ExplorerTree.ExpandShortcutImage as Long

Retrieves or sets a value that indicates the index of the image being displayed to expand the shortcuts.

Type	Description
Long	A Long expression that indicates the index of the icon being used to display the expand button in the control's shortcut bar.

By default, The ExpandShortcutImage property is 0. If the ExpandShortcutImage property is 0, the control displays the default icon to show the expand button in the shortcut bar. If the ExpandShortcutImage property is greater than 0, it indicates the index of the icon being used to display the expand button. The [Images](#) method assigns a collection of icons to the control. The expand button in the shortcut bar is hidden, if the ExpandShortcutImage property is -1. Use the [AllowResizeShortcutBar](#) property to enable or disable resizing the shortcut bar. Use the [ExpandShortcutCount](#) property to specify the number of shortcuts being expanded. An expanded shortcut displays its full caption, as [Shortcut](#) property specifies.

property ExplorerTree.FocusGroup as Group

Retrieves the group that has the focus.

Type	Description
Group	A Group object that has the focus.

Use the FocusGroup property to get the group that has the focus. Use the FocusGroup property determines the group where the drag and drop operations beings. The [OLEStartDrag](#) event notifies your application that the user starts dragging data from the control.

property ExplorerTree.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object that specifies the control's font.

Use the Font property to specify the control's font. Use properties like Font, Bold, Italic or Underline to specify the font's attribute while painting an object like group or item. The Font property assign the [Font](#) properties for all groups in the control.

property ExplorerTree.ForeColor as Color

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

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

Use the ForeColor property to specify the control's foreground color. Use the [ForeColorGroup](#) property to specify the default foreground color used to paint the groups captions. Use the [ForeColor](#) property to specify the foreground color for group's caption, Use the [ForeColorList](#) property to specify the foreground color of the group's list. Use the [ItemForeColor](#) property to specify the item's foreground color.

property ExplorerTree.ForeColorGroup as Color

Retrieves or sets a value that indicates the default group's foreground color.

Type	Description
Color	A color expression that indicates the default foreground color for group captions.

Use the ForeColorGroup property to specify the foreground color for all groups. The ForeColorGroup property changes only the foreground color of group captions. Use the [ForeColor](#), [ForeColorList](#) property to change the foreground color for a specific group.

property ExplorerTree.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

Type	Description
New as Boolean	A 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.

property ExplorerTree.GroupAppearance as AppearanceEnum

Specifies the group's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the group's appearance.

Use the GroupAppearance to specify the group's appearance.

property ExplorerTree.GroupFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Group

Gets the group 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 X location of the mouse pointer. The x values is always expressed in client coordinates
Group	A Group object over the point (X, Y).

Use the GroupFromPoint property to get the group from the point specified by the {X,Y}. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the GroupFromPoint property determines the group from the cursor.** Use the [ItemFromPoint](#) property to get the cell or item from the cursor. Use the [ColumnFromPoint](#) property to access the column over the point. Use the GroupFromPoint property to get the group's caption from the cursor. Use the [GroupListFromPoint](#) property to get the group's list from cursor.

The following VB sample displays the group's caption from the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim g As EXPLORERTREELibCtl.Group
    With ExplorerTree1
        Set g = .GroupListFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not (g Is Nothing) Then
            Debug.Print g.Caption
        End If
    End With
End Sub
```

The following VFP sample displays the group's caption from the cursor:

```
with thisform.Olecontrol1
    local g
```

```
g = .GroupFromPoint(-1,-1)
If !isnull(g) Then
  with g
    wait window .Caption nowait
  endwith
EndIf
endwith
```

property ExplorerTree.GroupHeight as Long

Specifies the group's height.

Type	Description
Long	A long expression that indicates the height of group's caption.

Use GroupHeight property to specify the group's caption height in pixels.

property ExplorerTree.GroupListFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Group

Gets the group's list 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.
Group	A Group object whose list is under the cursor.

Use the GroupFromPoint property to get the group's list from the point specified by the {X,Y}. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the GroupListFromPoint property determines the group's list from the cursor.** Use the [ItemFromPoint](#) property to get the cell or item from the cursor. Use the [ColumnFromPoint](#) property to access the column over the point. Use the [GroupFromPoint](#) property to get the group's caption from the cursor. Use the [ShortcutFromPoint](#) property to retrieve the shortcut from the cursor.

The following VB sample prints the cell over the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim g As EXPLORERTREELibCtl.Group
    With ExplorerTree1
        Set g = .GroupListFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not (g Is Nothing) Then
            With g
                Dim h As Long, c As Long, hit as Long
                h = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, hit)
                If Not (h = 0) Then
                    With g.Items
                        Debug.Print .CellCaption(h, c)
                    End With
                End If
            End With
        End With
    End Sub
```

```
End If
End With
End Sub
```

The following VFP sample prints the cell's caption from the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform.Olecontrol1
  local g
  g = .GroupListFromPoint(-1,-1)
  If !isnull(g) Then
    with g
      local h, c, hit
      c = 0
      hit = 0
      h = .ItemFromPoint(-1,-1,@c,@hit)
      If h # 0 Then
        with .Items
          .DefaultItem = h
          wait window .CellCaption(0, c) nowait
        endwith
      EndIf
    endwith
  EndIf
endwith
```

property ExplorerTree.Groups as Groups

Retrieves the control's groups collection.

Type	Description
Groups	A Groups object that indicates the control's groups collection.

Use the Groups property to access the control's groups collection. Use the [Items](#) property to access the items collection of the group. Use the [Columns](#) property to access the group's collection of columns. The [Caption](#) property specifies the group's caption, while the [CellCaption](#) property indicates the caption of a specified cell.

The following VBA sample enumerates all groups and items being shown on the control:

```
Dim e As EXPLOREERTREELib.ExplorerTree
Set e = ExplorerTree1.Object
With e
    Dim g As EXPLOREERTREELib.Group
    For Each g In .Groups
        Debug.Print "Enumerate Group " & g.Caption
        For Each i In g.Items
            With g.Items
                Debug.Print "Item " & .CellCaption(i, 0)
            End With
        Next
    Next
Next
End With
```

The following VBA sample enumerates all groups and items being checked on the control:

```
Dim e As EXPLOREERTREELib.ExplorerTree
Set e = ExplorerTree1.Object
With e
    Dim g As EXPLOREERTREELib.Group
    For Each g In .Groups
        Debug.Print "Enumerate Group " & g.Caption
        For Each i In g.Items
            With g.Items
                If Not (.CellState(i, 0) = 0) Then
```

```
Debug.Print "Item " & .CellCaption(i, 0)
```

```
End If
```

```
End With
```

```
Next
```

```
Next
```

```
End With
```

property ExplorerTree.HandCursor as Boolean

Specifies whether the control uses the hand cursor when the mouse is over the group.

Type	Description
Boolean	A boolean expression that specifies whether the control uses the hand cursor when the mouse is over the group.

By default, the HandCursor property is True.

property ExplorerTree.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	<p>The HTMLPicture specifies the picture being associated to a key. It can be one of the followings:</p> <ul style="list-style-type: none">• 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), <p>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</p>

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the 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 "pic1" 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 ExplorerTree.hWnd as Long

Retrieves the handle of the control's window.

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

Specifies the handle of the control's window used by API functions. Use the group's [hWnd](#) property to get the handle of the group's list window.

method ExplorerTree.Images (Handle as Variant)

Sets the control's handle image list.

Type

Description

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 `lVal` 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

Handle as Variant

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 control's image holder. The [ImageSize](#) property defines the size (width/height) of the icons within the control's Images collection. Use the [ShowImageList](#) property to display the control's images panel. The following sample uses the Microsoft Image List control:

ExplorerTree1.Images ImageList1.hImageList

The following sample adds two icons to the control's images collection using the base64 encoding strings:

```
ExplorerTree1.Images
```

```
"gBJJgBAICAAGAAEAQAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmEx  
mUzmk1m03nE5nU7nk9n0/oFBoVDolFo1HpFJpVLpInp1PqFRqVTqIVq1XrFZrVbrldr1fsFhs  
Vjslls1ntFptVrtltt1vuFxuVzul1u13vF5vV7vI9v1/wGBwWDwCAw0Tf9dYGLAGLYGEvsQxGNx  
mSqmOymPzGYyFvwyAyeSzGiysPpObxmozWlxOWztlYwKx+x02j2usoOcjWqzOh020xOvr  
Of32J23G3GyyfHmu6jPO3vM4nA4vCqPU5nK12gjOg7WmmnEjfQ7O/03E6fm4PWpXow/f  
5Gz8AA0HM7uTmmY8W7xnE6T3vU5bGPYpz1vW9zvNu+UDuq+7XJk3j9ue5MAwRBsCKp  
AyNQs5T7wc+afN46SNOpDsMK/C0OQU/CjxK88NxJAMTrPFLDwk68AxNGa4QsrjqR3IEgy  
Flcil0iwfJXJCPJAKSSAAkqUSgnEHyKmCKI2eZ/yyjZwH/LsuB+cAfvGB5gAe8YBmAAaNkAA  
M2zZN0plzNc1o3Ok2AOQADy4A5wT2jR8AfQKNn8B9Co2f5P0TREHoCA= ="
```



property ExplorerTree.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 (`number` ex-html tag, [CellImage](#), [CellImages](#))
- check-box or radio-buttons ([CellHasCheckBox](#), [CellHasRadioButton](#))
- expand/collapse glyphs ([HasButtons](#), [HasButtonsCustom](#))
- header's sorting or drop down-filter glyphs

method `ExplorerTree.OLEDrag ()`

Causes a component to initiate an OLE drag/drop operation.

Type	Description
------	-------------

Only for internal use.

property ExplorerTree.OLEDropMode as exOLEDropModeEnum

Returns or sets how a target component handles drop operations

Type	Description
exOLEDropModeEnum	An exOLEDropModeEnum expression that indicates the OLE Drag and Drop mode.

By default, the OLEDropMode property is exOLEDropNone. Currently, the ExExplorerTree control supports only manual OLE Drag and Drop operation. See the [OLEStartDrag](#) and [OLEDragDrop](#) events for more details about implementing drag and drop operations into the ExExplorerTree control.

property ExplorerTree.Picture as IPictureDisp

Retrieves or sets a graphic to be displayed in the control's background.

Type	Description
IPictureDisp	A Picture object that identifies the control's picture.

The control's picture is displayed on the control's background. Use the [PictureDisplay](#) property to specify the way how the picture is arranged on the control's background. Use the group's [Picture](#) property to specify a picture for a given group.

property ExplorerTree.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 arranged on the control's background.

By default, the PictureDisplay is exTile. The PictureDisplay property has no effect if the control's Picture is empty.

method `ExplorerTree.Refresh ()`

Refreshes the control.

Type	Description
------	-------------

Use the `Refresh` method to force repainting the control.

method ExplorerTree.Replacelcon ([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	<p>A Variant expression that specifies the icon to add or insert, as one of the following options:</p> <ul style="list-style-type: none">• a long expression that specifies the handle of the icon (HICON)• a string expression that indicates the path to the picture file• a string expression that defines the picture's content encoded as BASE64 strings using the eXImages tool• a Picture reference, which is an object that holds image data. It is often used in controls like PictureBox, Image, or in custom controls (e.g., IPicture, IPictureDisp) <p>If the Icon parameter is 0, it specifies that the icon at the given Index is removed. Furthermore, setting the Index parameter to -1 removes all icons.</p> <p>By default, if the Icon parameter is not specified or is missing, a value of 0 is used.</p>
Index as Variant	<p>A long expression that defines the index of the icon to insert or remove, as follows:</p> <ul style="list-style-type: none">• A zero or positive value specifies the index of the icon to insert (when Icon is non-zero) or to remove (when the Icon parameter is zero)• A negative value clears all icons when the Icon parameter is zero <p>By default, if the Index parameter is not specified or is missing, a value of -1 is used.</p>
Return	Description
Long	A long expression that indicates the index of the icon in the images collection

Use the Replacelcon property to add, remove or replace an icon in the control's images

collection. Also, the `Replacelcon` property can clear the images collection. Use the `Images` method to attach an image list to the control.

The following sample shows how to add a new icon to control's images list:

```
i = ExplorerTree1.Replacelcon( LoadPicture("d:\icons\help.ico").Handle), where i is the index to insert the icon
```

The following sample shows how to replace an icon into control's images list::

```
i = ExplorerTree1.Replacelcon( LoadPicture("d:\icons\help.ico").Handle, 0), in this case the i is zero, because the first icon was replaced.
```

The following sample shows how to remove an icon from control's images list:

```
ExplorerTree1.Replacelcon 0, i, in this case the i must be the index of the icon that follows to be removed
```

The following sample shows how to clear the control's icons collection:

```
ExplorerTree1.Replacelcon 0, -1
```

property ExplorerTree.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 [ScrollBars](#) property to specify which scroll bar is visible or hidden in the control. 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 ExplorerTree.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 [ScrollBars](#) property to specify which scroll bar is visible or hidden in the control. 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 ExplorerTree.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 [ScrollPartCaption](#) 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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar.

property ExplorerTree.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 [ScrollBars](#) property to specify which scroll bar is visible or hidden in the control. 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 ExplorerTree.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 fourth 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 fourth 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.

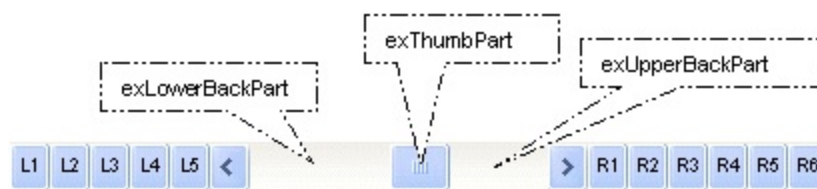
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

property ExplorerTree.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 displayed
String	A String expression that specifies the caption being displayed on the part of the scroll bar.

Use the ScrollPartCaption 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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar. Use the [ScrollFont](#) property to specify the font in the control's scroll bar.



By default, the following parts are shown:

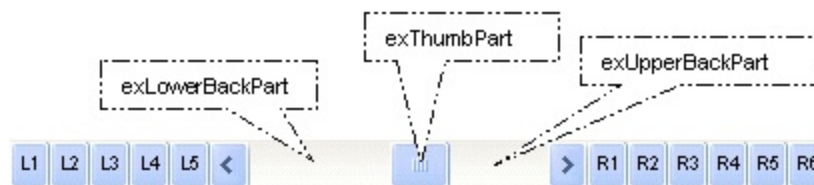
- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right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

property ExplorerTree.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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [ScrolPartCaption](#) property to specify the caption of the scroll's part. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar.

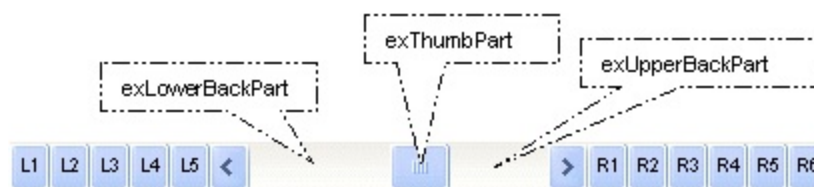


property ExplorerTree.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [ScrolPartCaption](#) property to specify the caption of the scroll's part. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar. Use the [Background](#) property to change the visual appearance for any part in the control's 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)

property ExplorerTree.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](#)(exVSTThumb) or [Background](#)(exHSTThumb) 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. Use the [ScrollThumbSize](#) property to define a fixed size for the scrollbar's thumb.

property ExplorerTree.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. The [OffsetChanged](#) event notifies your application that the user changes the scroll position. Use the [SortPartVisible](#) property to specify the parts being visible in the control's scroll bar. Use the [ScrollBars](#) property to specify the visible scrollbars in the control.

property ExplorerTree.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 [ScrollBars](#) property to specify which scroll bar is visible or hidden in the control. 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 ExplorerTree.SelectShortcut as Variant

Selects and displays the specified shortcut.

Type	Description
Variant	A String expression that indicates the caption of the Shortcut being selected.

The SelectShortcut property indicates the shortcut being selected. The [Shortcut](#) property of the Group object indicates the shortcuts that may be selected. The Group objects with the same Shortcut property indicates a set of groups, that may be selected using the SelectShortcut property. The [SelectShortcut](#) event is fired when the user clicks a shortcut or when the user calls the SelectShortcut property. The [ShowShortcutBar](#) property shows or hides the control's shortcut bar. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar. The [ShortcutBarHeight](#) property sets or gets a value that indicates the height in pixels of the control's shortcut bar

property ExplorerTree.ShortcutBarBackColor as Color

Retrieves or sets the shortcut bar's background color.

Type	Description
Color	A color expression that indicates the shortcut'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.

Use the `ShortcutBarBackColor` property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutBarSelBackColor](#) property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. The [ShortcutBarSelCaptionBackColor](#) property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed. The [ShortcutResizeBackColor](#) property defines the visual appearance/background color slider that resizes the shortcut bar.

property ExplorerTree.ShortcutBarHeight as Long

Selects and displays the specified shortcut.

Type	Description
Long	A long expression that indicates the height of the shortcut bar.

By default, the `ShortcutBarHeight` property is 24 pixels. The `ShortcutBarHeight` property defines the height for each item in the shortcut bar. For instance, if the shortcut bar has no expanded shortcuts, the `ShortcutBarHeight` property defines the height of the shortcut bar, that displays a single item where all shortcuts are displayed. Use the [ShowShortcutBar](#) property to show or hide the control's shortcut bar. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar.

property ExplorerTree.ShortcutBarSelBackColor as Color

Retrieves or sets the background color for the selected icon in the shortcut bar.

Type	Description
Color	A color expression that indicates the background color for the selected shortcut. 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 `ShortcutBarSelBackColor` property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. The [ShortcutBarSelCaptionBackColor](#) property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed. Use the [ShortcutBarBackColor](#) property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutResizeBackColor](#) property defines the visual appearance/background color slider that resizes the shortcut bar.

property ExplorerTree.ShortcutBarSelCaptionBackColor as Color

Retrieves or sets the background color for selected shortcut when its entire caption is displayed.

Type	Description
Color	A color expression that indicates the background color for the selected shortcut. 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 `ShortcutBarSelCaptionBackColor` property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed. The [ShortcutBarSelBackColor](#) property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. Use the [ShortcutBarBackColor](#) property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutResizeBackColor](#) property defines the visual appearance/background color slider that resizes the shortcut bar.

property ExplorerTree.ShortcutFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as String

Gets the shortcut from the cursor.

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 shortcut from the point.

Use the `ShortcutFromPoint` property to get the shortcut from the point specified by the {X,Y}. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the `ShortcutFromPoint` property determines the shortcut from the cursor.** Use the [ItemFromPoint](#) property to get the cell or item from the cursor. Use the [ColumnFromPoint](#) property to access the column over the point. Use the [GroupFromPoint](#) property to get the group's caption from the cursor. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same `Shortcut` property are displayed in the same shortcut. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar. Use the [ShowToolTip](#) property to display programmatically a custom tooltip.

The following VB sample displays the shortcut from the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Debug.Print ExplorerTree1.ShortcutFromPoint(-1, -1)
End Sub
```

The following VB.NET sample displays the shortcut from the cursor:

```
Private Sub AxExplorerTree1_MouseMoveEvent(ByVal sender As Object, ByVal e As AxEXPLORERTREELib._IExplorerTreeEvents_MouseMoveEvent) Handles AxExplorerTree1.MouseMoveEvent
    Debug.Print(AxExplorerTree1.get_ShortcutFromPoint(-1, -1))
End Sub
```

The following C# sample displays the shortcut from the cursor:

```
private void axExplorerTree1_MouseMoveEvent(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_MouseMoveEvent e)
{
    System.Diagnostics.Debug.WriteLine(axExplorerTree1.get_ShortcutFromPoint(-1, -1));
}
```

The following C++ sample displays the shortcut from the cursor:

```
void OnMouseMoveExplorertree1(short Button, short Shift, long X, long Y)
{
    OutputDebugString( m_explorerTree.GetShortcutFromPoint( -1, -1 ) );
}
```

The following VFP sample displays the shortcut from the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

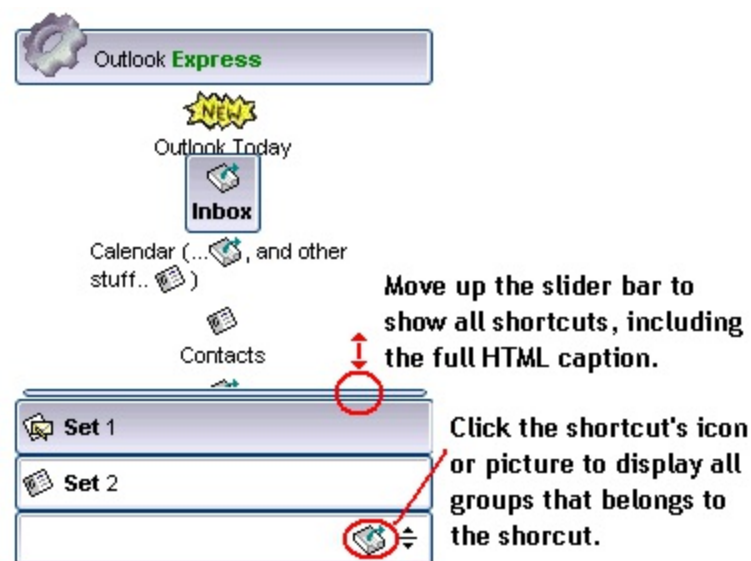
with thisform.ExplorerTree1
    wait window nowait .ShortcutFromPoint(-1,-1)
endwith
```

property ExplorerTree.ShortcutPicture(Shortcut as String) as Variant

Specifies a custom-size picture assigned to a shortcut.

Type	Description
Shortcut as String	A String expression that indicates the caption of the shortcut where a custom size picture is assigned.
Variant	A String expression that indicates the path to the picture file or 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 being assigned to the shortcut.

Use the ShortcutPicture property to assign a custom size picture to a shortcut. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The shortcut bar displays the first icon found in the HTML caption, if a custom size picture is not assigned to the shortcut using the ShortcutPicture property. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar. Use the [ShortcutBarHeight](#) property to define the height in pixels of one shortcut items in the shortcut bar. Use the [ExpandShortcutCount](#) property to expand the number of shortcuts in the control's shortcut bar. The ShortcutPicture property has no effect if the shortcut bar is not visible, or there is no Group assigned to the specified shortcut.



The following VB sample assign a custom size picture to the shortcut named "`1 Set 1`":

With ListBar1

```
.ShortcutBarHeight = 38
```


.ShortcutPicture(" 1 Set 1") = "D:\Temp\Icons\misc.gif"
End With

The following screen shot shows the shortcutbar when there is no items expanded:



The following screen shot shows the shortcutbar when there is a single shortcut expanded (**Set 1**)



property ExplorerTree.ShortcutPictureHeight as Long

Specifies the height in pixels of the custom size picture being displayed in the shortcut bar.

Type	Description
Long	A Long expression that indicates the height of the picture to be stretched to.

By default, the `ShortcutPictureHeight` property is -1. If the `ShortcutPictureHeight` property is -1, the shortcut's picture is not stretched on the height. If the `ShortcutPictureHeight` property is positive, it indicates the height in pixels to be stretched to. The `ShortcutPictureHeight` property specifies the height of the picture when assigning a custom size picture using the [ShortcutPicture](#) property. The [ShortcutPictureWidth](#) property specifies the width in pixels of the picture to be stretched to. Use the [ShortcutBarHeight](#) property to specify the height in pixels of the control's shortcut bar. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same `Shortcut` property are displayed in the same shortcut. The shortcut bar displays the first icon found in the HTML caption, if a custom size picture is not assigned to the shortcut using the `ShortcutPicture` property.

property ExplorerTree.ShortcutPictureWidth as Long

Specifies the width in pixels of the custom size picture being displayed in the shortcut bar.

Type	Description
Long	A long expression that indicates the width of the picture to be stretched to.

By default, the `ShortcutPictureWidth` property is `-1`. If the `ShortcutPictureWidth` property is `-1`, the shortcut's picture is not stretched on the width. If the `ShortcutPictureWidth` property is positive, it indicates the width in pixels to be stretched to. The `ShortcutPictureWidth` property specifies the width of the picture when assigning a custom size picture using the [ShortcutPicture](#) property. The [ShortcutPictureHeight](#) property specifies the height in pixels of the picture to be stretched to. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same `Shortcut` property are displayed in the same shortcut. The shortcut bar displays the first icon found in the HTML caption, if a custom size picture is not assigned to the shortcut using the `ShortcutPicture` property.

property ExplorerTree.ShortcutResizeBackColor as Color

Retrieves or sets the background color for the shortcut's resize bar.

Type	Description
Color	A color expression that indicates the background color for the slider that resizes the shortcut bar. 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 `ShortcutResizeBackColor` property defines the visual appearance/background color slider that resizes the shortcut bar. Use the [ShortcutBarBackColor](#) property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutBarSelBackColor](#) property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. The [ShortcutBarSelCaptionBackColor](#) property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed.

property ExplorerTree.ShowFocusRect as Boolean

Specifies a value that indicates whether the control draws the focused item.

Type	Description
Boolean	A boolean expression that indicates whether the control's focused item is marked.

Currently the ShowFocusRect property is not available. Use the [ShowFocusRect](#) property to mark the focused item inside a group.

property ExplorerTree.ShowImageList as Boolean

Retrieves or sets a value that indicates whether the image list window is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the control's images list window is visible or hidden.

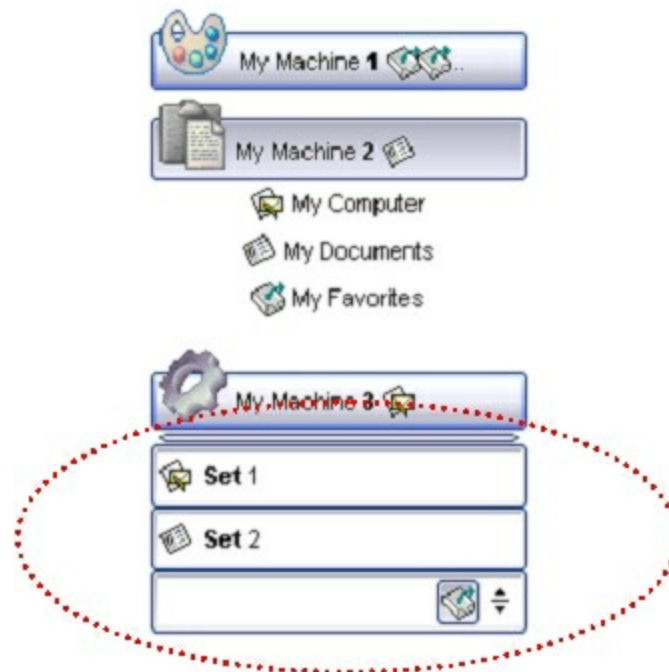
The ShowImageList control has no effect at runtime. It has effect only at design time. Use the method to change the control's images list collection, or use [Replacelcon](#) method to add, remove, or clear the images collection. Use the [Images](#) method to assign a list of images to the control.

property ExplorerTree.ShowShortcutBar as Boolean

Retrieves or sets a value that indicates whether the image shortcut bar is visible or hidden.

Type	Description
Boolean	A Boolean expression that indicates whether the control's shortcut bar is visible or hidden.

By default, the ShowShortcutBar property is False, and that means that the shortcut bar is hidden. The shortcut bar if visible, it is displayed on the bottom side of the control as seen in the following screen shot. The Shortcut feature allows you to group the groups in sets, so you may have sets that contains groups, and groups that contains items. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar. The [ShortcutBarHeight](#) property sets or gets a value that indicates the height in pixels of the control's shortcut bar. The [ShortcutBarBackColor](#) property indicates the shortcut bar's background color or its visual appearance if using skins. The [ShortcutResizeBackColor](#) property changes the visual appearance of the resizing bar of the shortcut bar. The [SelectShortcut](#) property selects a shortcut. When a shortcut is selected, the control displays only groups with the [Shortcut](#) property as being the SelectShortcut property.



The red circle marks the control's shortcut bar. Use the [AllowResizeShortcutBar](#) property to specify whether the user may expand the shortcut bar using the mouse. Use the [ExpandShortcutImage](#) property to display a custom icon in the right side expand button. Use the [ExpandShortcutCount](#) property to specify the number of shortcuts that display their

full caption, else the first icon in the caption is displayed or the assigned picture is displayed. The [ExpandShortcut](#) event notifies your application when the user resizes the control's shortcut bar. The control fires the [SelectShortcut](#) event when the user selects a shortcut, so groups that belongs to the shortcut are displayed.

method ExplorerTree.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	<p>The ToolTip parameter can be any of the following:</p> <ul style="list-style-type: none">• 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	<p>The Title parameter can be any of the following:</p> <ul style="list-style-type: none">• 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	<p>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.</p> <p>The Alignment parameter can be one of the following:</p> <ul style="list-style-type: none">• 0 - exTopLeft• 1 - exTopRight• 2 - exBottomLeft• 3 - exBottomRight• 0x10 - exCenter• 0x11 - exCenterLeft• 0x12 - exCenterRight• 0x13 - exCenterTop• 0x14 - exCenterBottom <p>By default, the tooltip is aligned relative to the top-left corner (0 - exTopLeft).</p>

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)

X 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)

Y as Variant

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**
- `<i> ... </i>` 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.

The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using `<a ;exp=>` or `<a ;e64=>` anchor tags. The `exp/e64` field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

- `exp`, stores the plain text to be shown once the user clicks the anchor, such as "`<a ;exp=show lines>`"
- `e64`, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "`<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY`" that displays `show lines-` in gray when the user clicks the `+` anchor. The "`gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY`" string encodes the "`<fgcolor 808080>show lines<a>-</fgcolor>`" The `Decode64Text/Encode64Text` methods of the `eXPrint` can be used to decode/encode `e64` fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "`<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3`" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the `+` sign.

- ** ... ** 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=rrggb> ... </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=rrggb> ... </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=rrggb> ... </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=rrggb> ... </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
- **number[:width]** 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.
- **key[:width]** 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 **&**; (&), **<**; (<), **>**; (>), **"**; (") 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 FFFFFFFF;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 ExplorerTree.Template as String

Specifies the control's template.

Type	Description
String	A string expression that indicates the initialization code.

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 ExplorerTree.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 ExplorerTree.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 use constant expressions as follows:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may start with 0x which indicates a hexa decimal representation, else it should start with a 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 indicate 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 ExplorerTree.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. Use the [ToolTip](#) property to specify a tooltip to be shown when the cursor hovers the group's caption. Use the [CellToolTip](#) property to specify the cell's tooltip. Use the [ShowToolTip](#) method to display a custom tooltip.

property ExplorerTree.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. Use the [CellToolTip](#) property to specify the cell's tooltip. Use the [ToolTip](#) property to specify a tooltip to be shown when the cursor hovers the group's caption. Use the [ShowToolTip](#) method to display a custom tooltip. You can use the HTML element to specify a different font for portions of text in your tooltip.

property ExplorerTree.ToolTipMargin as String

Defines the size of the control's tooltip margins.

Type	Description
String	<p>A string expression that defines the horizontal and vertical margins (separated by comma) of the control's tooltip as one of the following formats:</p> <ul style="list-style-type: none">• "value", where value is a positive number, that specifies the horizontal and vertical margins, such as "4" equivalent of "4,4"• "value,", where value is a positive number, that specifies the horizontal margin, such as "4," equivalent of "4,0"• ",value", where value is a positive number, that specifies the vertical margin, such as ",4" equivalent of "0,4"• "horizontal,vertical", where horizontal and vertical are positive numbers, that specifies the horizontal and vertical margins, such as "4,4"

By default, the size of the tooltip margin is "4" (horizontal and vertical). For instance, `ToolTipMargin = "8"` changes the horizontal and vertical margins are set to 8 pixels. `ToolTipMargin = "8,4"` changes the horizontal margin to 8 pixels and the vertical margin to 4 pixels. The [ToolTipWidth](#) property specifies a value that indicates the width of the tooltip window, in pixels. Use the [ShowToolTip](#) method to display a custom 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. The [ToolTipDelay](#) property specifies the time in ms that passes before the ToolTip appears.

property ExplorerTree.ToolTipPopDelay as Long

Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.

Type	Description
Long	A long expression that specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The [ToolTipDelay](#) property specifies the time in ms that passes before the ToolTip appears. Use the [ToolTipWidth](#) property to specify the width of the tooltip window. Use the [ToolTipFont](#) property to assign a font for the control's tooltip. Use the [Background\(exToolTipAppearance\)](#) property indicates the visual appearance of the borders of the tooltips. Use the [Background\(exToolTipBackColor\)](#) property indicates the tooltip's background color. Use the [Background\(exToolTipForeColor\)](#) property indicates the tooltip's foreground color. Use the [CellToolTip](#) property to specify the cell's tooltip. Use the [ToolTip](#) property to specify a tooltip to be shown when the cursor hovers the group's caption. Use the [ShowToolTip](#) method to display a custom tooltip.

property ExplorerTree.ToolTipWidth as Long

Specifies a value that indicates the width of the tooltip window, in pixels.

Type	Description
Long	A long expression that indicates the width of the tooltip window in pixels.

Use the `ToolTipWidth` property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. The [ToolTipPopDelay](#) property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The [ToolTipDelay](#) property specifies the time in ms that passes before the ToolTip appears. Use the [ToolTipFont](#) property to assign a font for the control's tooltip. Use the [Background\(exToolTipAppearance\)](#) property indicates the visual appearance of the borders of the tooltips. Use the [Background\(exToolTipBackColor\)](#) property indicates the tooltip's background color. Use the [Background\(exToolTipForeColor\)](#) property indicates the tooltip's foreground color. Use the [CellToolTip](#) property to specify the cell's tooltip. Use the [ToolTip](#) property to specify a tooltip to be shown when the cursor hovers the group's caption. Use the [ShowToolTip](#) method to display a custom tooltip.

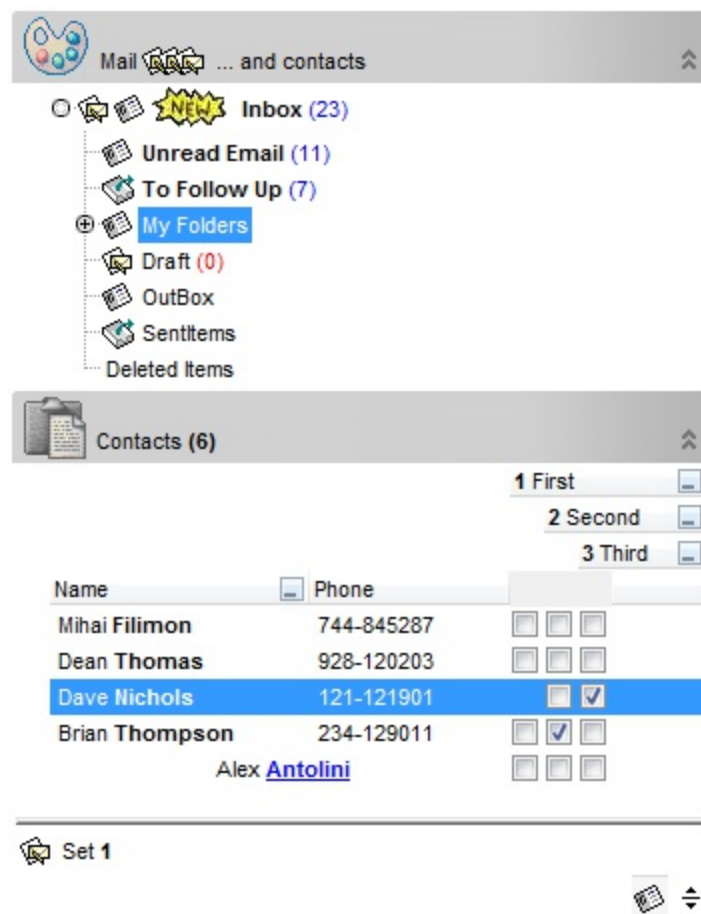
property ExplorerTree.UseVisualStyle as UIVisualThemeEnum

Specifies whether the control uses the current visual theme to display certain UI parts.

Type	Description
UIVisualStyleEnum	An UIVisualThemeEnum expression that specifies which UI parts of the control are shown using the current visual theme.

By default, the UseVisualStyle property is exDefaultVisualStyle, which means that all known UI parts are shown as in the current theme. The UseVisualStyle property may specify the UI parts that you need to enable or disable the current visual theme. The UI Parts are like header, filterbar, check-boxes, buttons and so on. The UseVisualStyle property has effect only a current theme is selected for your desktop. The UseVisualStyle property. Use the [Appearance](#) property of the control to provide your own visual appearance using the EBN files.

The following screen shot shows the control while the UseVisualStyle property is exDefaultVisualStyle:



since the second screen shot shows the same data as the UseVisualStyle property is exNoVisualStyle:

Mail ... and contacts ^

- NEW** **Inbox (23)**
 - Unread Email (11)**
 - To Follow Up (7)**
 - My Folders**
 - Draft (0)**
 - OutBox**
 - SentItems**
 - Deleted Items**

Contacts (6) ^

1 First v

2 Second v

3 Third v

Name v	Phone		
Mihai Filimon	744-845287	<input type="checkbox"/>	<input type="checkbox"/>
Dean Thomas	928-120203	<input type="checkbox"/>	<input type="checkbox"/>
Dave Nichols	121-121901	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Brian Thompson	234-129011	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Alex Antolini		<input type="checkbox"/>	<input type="checkbox"/>

Set 1



property ExplorerTree.Version as String

Retrieves the control's version.

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

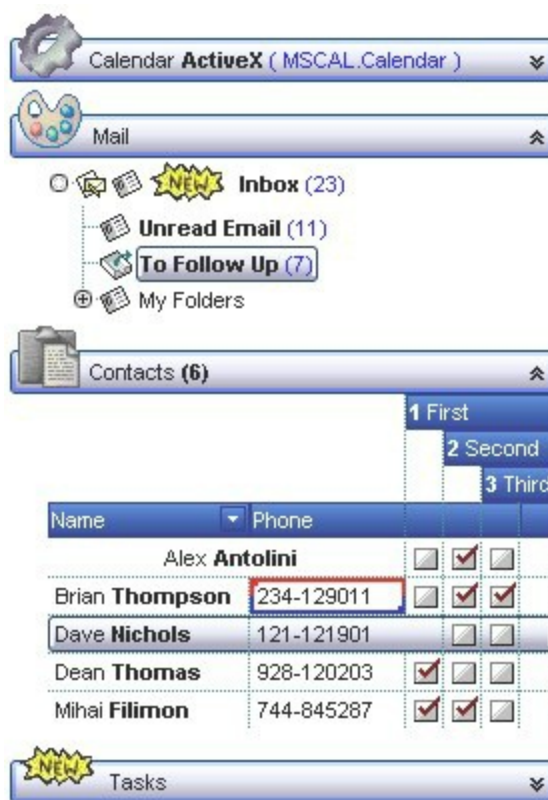
For instance, the unicode version of the demo control could be 1.0.2.2.DEMO.UNICODE.

property ExplorerTree.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 its simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part.



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

- control's **header bar**, [BackColorHeader](#) property
- control's **filter bar**, [FilterBarBackColor](#) property
- **selected item** or cell, [SelBackColor](#) property
- **item**, [ItemBackColor](#) property
- **cell**, [CellBackColor](#) property
- cell's **button**, "drop down" filter bar button, "close" filter bar button, and so on, [Background](#) property
- and so on.

Group object

The Group object holds a collection of columns and items that can be displayed as a tabular view or hierarchical view. Use the [Add](#) method to [Groups](#) collection to add new groups to the control. Use the [Items](#) property to access the group's [Items](#) collection. Use the [Columns](#) property to access the group's [Columns](#) collection. Use the [AddItem](#) method to add new items to the group. Use the [Add](#) method to add new columns to the group.

Name	Description
Alignment	Retrieves or sets a value that indicates the caption's alignment.
AllowEdit	Gets or sets a value indicating whether edits are allowed.
AllowExpand	Specifies whether the group can be expanded or collapsed.
AllowScroll	Enables or disables scrolling the group when it is expanded or collapsed.
ApplyFilter	Applies the filter.
ASCIILower	Specifies the set of lower characters.
ASCIIUpper	Specifies the set of upper characters.
AutoHeight	Specifies a value that indicates whether the height of the group's list is computed based on the visible items in the group.
AutoSearch	Enables or disables the group's incremental searching feature.
BackColor	Retrieves or sets the group's background color.
BackColor2	Specifies the color at the ending boundary line of the gradient group's caption.
BackColorAlternate	Specifies the background color used to display alternate items in the group.
BackColorHeader	Specifies the group header's background color.
BackColorLevelHeader	Specifies the multiple levels header's background color.
BackColorList	Retrieves or sets a value that indicates the background color of the list when the group is active.
BackColorLock	Retrieves or sets a value that indicates the group's background color for the locked area.
BeginUpdate	Maintains performance when items are added to the group one at a time. This method prevents the control from

painting until the EndUpdate method is called.

[Bold](#) Specifies a value that indicates whether the group's caption should appear in bold.

[BorderColor](#) Specifies the color of group's border.

[Caption](#) Specifies the group's caption.

[CaptionFormat](#) Specifies how the group's caption is displayed.

[CheckImage](#) Retrieves or sets a value that indicates the image used by cells of checkbox type.

[ClearFilter](#) Clears the filter.

[ColumnAutoResize](#) Returns or sets a value indicating whether the group will automatically size its visible columns to fit on the group's list width.

[ColumnFromPoint](#) Retrieves the column from point.

[Columns](#) Retrieves the group's columns 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 group scrolls columns pixel by pixel.

[CountLockedColumns](#) Retrieves or sets a value indicating the number of locked columns. A locked column is not scrollable.

[DataSource](#) Retrieves or sets a value that indicates the data source for the group.

[DefaultItemHeight](#) Retrieves or sets a value that indicates the default item height.

[Description](#) Changes descriptions for group objects.

[DrawGridLines](#) Retrieves or sets a value that indicates whether the grid lines are visible or hidden.

[EndUpdate](#) Resumes painting the group after painting is suspended by the BeginUpdate method.

[EnsureVisibleColumn](#) Scrolls the group's content to ensure that the column fits the client area.

[Expanded](#) Expands or collapses the group.

[ExpandOnDbClick](#) Specifies whether the item is expanded or collapsed if the user dbl clicks the item.

[ExpandOnKeys](#)

Specifies a value that indicates whether the control expands or collapses a node when user presses arrow keys.

[ExpandOnSearch](#)

Expands items automatically while user types characters to search for a specific item.

[FilterBarBackColor](#)

Specifies the background color of the group's filter bar.

[FilterBarCaption](#)

Specifies the filter bar's caption.

[FilterBarDropDownHeight](#)

Specifies the height of the drop down filter window proportionally with the height of the group's list.

[FilterBarFont](#)

Retrieves or sets the font for group's filter bar.

[FilterBarForeColor](#)

Specifies the foreground color of the group's filter bar.

[FilterBarHeight](#)

Specifies the height of the group's filter bar. If the value is less than 0, the filter bar is automatically resized to fit its description.

[FilterBarPrompt](#)

Specifies the caption to be displayed when the filter pattern is missing.

[FilterBarPromptColumns](#)

Specifies the list of columns to be used when filtering using the prompt.

[FilterBarPromptPattern](#)

Specifies the pattern for the filter prompt.

[FilterBarPromptType](#)

Specifies the type of the filter prompt.

[FilterBarPromptVisible](#)

Shows or hides the filter prompt.

[FilterCriteria](#)

Retrieves or sets the filter criteria.

[FilterInclude](#)

Specifies the items being included after the user applies the filter.

[Font](#)

Retrieves or sets the group's font.

[ForeColor](#)

Specifies the group's foreground color.

[ForeColorHeader](#)

Specifies the group header's foreground color.

[ForeColorList](#)

Retrieves or sets a value that indicates the foreground color of the group's list when it is active.

[ForeColorLock](#)

Retrieves or sets a value that indicates the group's foreground color for the locked area.

[FullRowSelect](#)

Enables full-row selection in the group.

[GetItems](#)

Gets the collection of items into a safe array,

[GridLineColor](#)

Specifies the group's grid line color.

GridLineStyle	Specifies the style for gridlines in the list part of the group.
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.
HasButtonsCustom	Specifies the index of icons for +/- signs when the HasButtons property is exCustom.
HasLines	Enhances the graphic representation of a group's hierarchy by drawing lines that link child items to their corresponding parent item.
HeaderAppearance	Retrieves or sets a value that indicates the appearance of the group's header.
HeaderHeight	Retrieves or sets a value indicating the group's header height.
HeaderSingleLine	Specifies whether the control resizes the columns header and wraps the captions in single or multiple lines.
HeaderVisible	Retrieves or sets a value that indicates whether the the group's header is visible or hidden.
Height	Specifies the height in pixels of the group's list.
HideSelection	Returns a value that determines whether selected item appears highlighted when the group loses the focus.
HotBackColor	Retrieves or sets a value that indicates the hot-tracking background color.
HotForeColor	Retrieves or sets a value that indicates the hot-tracking foreground color.
hWnd	Retrieves the group's window handle.
HyperLinkColor	Specifies the hyperlink color.
Image	Specifies group's image.
ImageAlignment	Specifies the icon's alignment.
Indent	Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
IndentGroupLeft	Specifies a value that indicates the indent of the group's list to the left side.
IndentGroupRight	Specifies a value that indicates the indent of the group's list to the right side.
IndentHeaderBottom	Specifies the number of pixels to indent the group's header from the bottom part.

IndentHeaderLeft	Specifies the number of pixels to indent the group's header from the left part.
IndentHeaderRight	Specifies the number of pixels to indent the group's header from the right part.
IndentHeaderTop	Specifies the number of pixels to indent the group's header from the top part.
Index	Retrieves the index of the object into the Groups collection..
Italic	Specifies a value that indicates whether the group's caption should appear in italic.
ItemFromPoint	Retrieves the item from point.
Items	Retrieves the group's items collection.
ItemsAllowSizing	Retrieves or sets a value that indicates whether a user can resize items at run-time.
Left	Specifies the distance between the left edge of the control and group's list.
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
Picture	Retrieves or sets a graphic to be displayed in the group's list.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the list's background
PictureDisplayLevelHeader	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's header background.
PictureLevelHeader	Retrieves or sets a graphic to be displayed in the control's header when multiple levels is on.
Position	Specifies the group's position.
PutItems	Adds an array of integer, long, date, string, double, float, or variant arrays to the group.
RadiolImage	Retrieves or sets a value that indicates the image used by cells of radio type.
RClickSelect	Retrieves or sets a value that indicates whether an item is selected using right mouse button.
Refresh	Refreshes the group's content.

[RightToLeft](#)

Indicates whether the group should draw right-to-left for RTL languages.

[ScrollBars](#)

Returns or sets a value that determines whether the group has horizontal and/or vertical scroll bars.

[ScrollButtonHeight](#)

Specifies the height of the button in the vertical scrollbar.

[ScrollButtonWidth](#)

Specifies the width of the button in the horizontal scrollbar.

[ScrollBySingleLine](#)

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.

[ScrollFont](#)

Retrieves or sets the scrollbar's font.

[ScrollPartCaption](#)

Specifies the caption being displayed on the specified scroll part.

[ScrollPartCaptionAlignment](#)

Specifies the alignment of the caption in the part of the scroll bar.

[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.

[SearchColumnIndex](#)

Retrieves or sets a value indicating the column's index for incremental searching feature.

[SelBackColor](#)

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

[SelBackMode](#)

Retrieves or sets a value that indicates whether the selection is transparent or opaque.

[SelectColumn](#)

Specifies whether the user selects cells only in SelectColumnIndex column, while FullRowSelect property is False.

[SelectColumnIndex](#)

Retrieves or sets a value that indicates the column's index where the user can select an item. It has effect only if FullRowSelect is false.

[SelectColumnInner](#)

Retrieves or sets a value that indicates the index of the inner cell that's selected.

[SelForeColor](#)

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

[SelLength](#)

Returns or sets the number of characters selected.

[SelStart](#)

Returns or sets the starting point of text selected; indicates the position of the insertion point if no text is selected.

[SetFocus](#)

Sets the keyboard focus to the group's list window.

[Shortcut](#)

Specifies the name of the shortcut which displays the group.

[ShowFocusRect](#)

Retrieves or sets a value indicating whether the group draws a thin rectangle around the focused item.

[ShowLockedItems](#)

Retrieves or sets a value that indicates whether the control displays the locked items.

[SingleSel](#)

Retrieves or sets a value that indicates whether the group supports single or multiple selection.

[SortOnClick](#)

Retrieves or sets a value that indicates whether the group sorts automatically the data when the user click on column's caption.

[StrikeOut](#)

Specifies a value that indicates whether the group's caption should appear in strikethrough.

[ToolTip](#)

Specifies the group's tooltip.

[Top](#)

Specifies the distance between the top edge of the control and group's list.

[TreeColumnIndex](#)

Retrieves or sets a value indicating the column's index where the hierarchy will be displayed.

[Underline](#)

Specifies a value that indicates whether the group's caption is underlined.

[UserData](#)

Specifies the group's extra data.

[Width](#)

Retrieves the width in pixels of the group's list.

property Group.Alignment as AlignmentEnum

Retrieves or sets a value that indicates the caption's alignment.

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

By default, the Alignment property is exCenter. The Alignment property doesn't align items in the group. Use the [Alignment](#) property to align a cell. If you want to align the entire list of items, you can handle the AddItem event where you can change the item's alignment each time when a new item is added to group.

property Group.AllowEdit as Boolean

Retrieves or sets a value that indicates whether the editing group is allowed or disabled.

Type	Description
Boolean	A boolean expression that indicates whether the editing group is allowed or disabled.

By default, the AllowEdit property is False. If the AllowEdit property is False, the events [BeforeCellEdit](#) and [AfterCellEdit](#) are not fired.

property Group.AllowExpand as Boolean

Specifies whether the group can be expanded or collapsed.

Type	Description
Boolean	A boolean expression that indicates whether the group can be expanded or collapsed.

Use the AllowExpand property to disable expanding or collapsing a group. Use the [Expanded](#) property to expand or collapse programmatically a group.

property Group.AllowScroll as Boolean

Enables or disables scrolling the group when it is expanded or collapsed.

Type	Description
Boolean	A boolean expression that specifies whether the control uses animation when a group is expanded or collapsed.

By default, the AllowScroll property is True.

method `Group.ApplyFilter ()`

Applies the filter.

Type	Description
------	-------------

The `ApplyFilter` method updates the group's content once that user sets the filter using the [Filter](#) and [FilterType](#) properties. Use the [ClearFilter](#) method to clear the group's filter

property Group.ASCIILower as String

Specifies the set of lower characters.

Type	Description
String	A string expression that indicates the set of lower characters used by auto search feature.

The ASCIILower and [ASCIIUpper](#) properties helps you to specify the set of characters that are used by the auto search feature. If you want to make the auto search feature case sensitive you have to use ASCIIUpper = "" . By default, the ASCIILower property is = "abcdefghijklmnopqrstuvwxyz?שבםףתס?גהאזחךטןמלפצע"

property Group.ASCIIUpper as String

Specifies the set of upper characters.

Type	Description
String	A string expression that indicates the set of upper characters used by auto search feature.

The [ASCIILower](#) and ASCIIUpper properties helps you to specify the set of characters that are used by the auto search feature. If you want to make the auto search feature case sensitive you have to use ASCIIUpper = "". By default, the ASCIIUpper property is = "ABCDEFGHIJKLMNOPQRSTUVWXYZÜÉÂĂŔŁÇĘĚĎÎĚÔÖŇÚŮÁÍÓÚŇ"

property Group.AutoHeight as Boolean

Specifies a value that indicates whether the height of the group's list is computed based on the visible items in the group.

Type	Description
Boolean	A boolean expression that that indicates whether the height of the group's list is computed based on the visible items in the group.

By default, the AutoHeight property is False. If the AutoHeight property is True, the group computes the [Height](#) property based on the visible [items](#) in the group's list. The vertical scroll bar of the group never shows up if the AutoHeight property is True. The exVertical flag is removed from the [Scrollbars](#) property, if the AutoHeight property is changed at runtime. You can call the ScrollBars property **after** setting the AutoHeight property, in case you need vertical scroll bar

property Group.AutoSearch as Boolean

Enables or disables the auto search feature.

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

By default, the AutoSearch property is True. The auto-search feature 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 group 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 group supports 'starts with' or 'contains' incremental search as described in the [AutoSearch](#) property of the [Column](#) object.

property Group.BackColor as Color

Retrieves or sets the group's background color.

Type	Description
Color	A color expression that indicates the background color of the group's caption. 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 specify the group's caption's background color. Use the [BackColorList](#) property to specify the background color for group's list. Use the [BackColorGroup](#) property to specify a default background color for groups. Use the group's [Picture](#) property to assign a picture the group's list.

property Group.BackColor2 as Color

Specifies the color at the ending boundary line of the gradient group's caption.

Type	Description
Color	A color expression that specifies the color at the ending boundary line of the gradient group's caption.

Use the BackColor and BackColor2 properties to display the group's caption using a gradient color.

property Group.BackColorAlternate as Color

Specifies the background color used to display alternate items in the group.

Type	Description
Color	A color expression that indicates the alternate background color.

By default, the group's BackColorAlternate property is zero. The group ignores the BackColorAlternate property if it is 0 (zero).

property Group.BackColorHeader as Color

Specifies the header's background color.

Type	Description
Color	A color expression that indicates the background color for the group's header.

Use the BackColorHeader and [ForeColorHeader](#) properties to customize the group's header. Use the [HeaderVisible](#) property to hide the group's header.

property Group.BackColorLevelHeader as Color

Specifies the multiple levels header's background color.

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

Use the BackColorHeader and [ForeColorHeader](#) properties to define colors used to paint the group's header bar. Use the BackColorLevelHeader property to specify the background color of the group's header bar when multiple levels are displayed. Use the [LevelKey](#) property to display the group'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 group's header displays multiple levels if there are two or more neighbor columns with the same non empty level key.

property Group.BackColorList as Color

Retrieves or sets a value that indicates the background color of the list when the group is active.

Type	Description
Color	A color expression that indicates the background color of the group's list.

The BackColorList property has effect only when the group is expanded. Use the [BackColor](#) property to specify the background color for the group's caption. Use the [BorderColor](#) property to specify the color of the group's list border. If you have not specified a the background color for the group's list using the BackColorList property, the control's [BackColor](#) property automatically is taken.

property Group.BackColorLock as Color

Retrieves or sets a value that indicates the group's background color for the locked area.

Type	Description
Color	A boolean expression that indicates the group's background color for the locked area.

The group may contains two kind of columns: locked and unlocked. The locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the [CountLockedColumns](#) to specify the number of locked columns. The unlocked contains the columns that can be scrolled horizontally. To change the background color of the group's unlocked area use [BackColor](#) property.

method Group.BeginUpdate ()

Maintains performance when items are added to the group one at a time.

Type

Description

This method prevents the group from painting until the EndUpdate method is called. The BeginUpdate and [EndUpdate](#) methods increases the speed of loading your items, by preventing painting the group when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too.

The following sample shows how to use the BeginUpdate and EndUpdate methods:

```
With ExplorerTree1.Groups.Add("Group 1")
  Set rs = CreateObject("ADODB.Recordset")
  rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
  .BeginUpdate
  .ColumnAutoResize = False
  .HeaderVisible = True
  ' By default, the group adds a default colum, so we remove it first
  .Columns.Clear
  For Each f In rs.Fields
    .Columns.Add f.Name
  Next
  .PutItems rs.GetRows()
  .EndUpdate
End With
```

property Group.Bold as Boolean

Specifies a value that indicates whether the group's caption should appear in bold.

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

Use the **Bold**, [Italic](#), [Underline](#) and [StrikeOut](#) properties to customize the caption's font attributes.

property Group.BorderColor as Color

Specifies the color of group's border.

Type	Description
Color	A color expression that specifies the color of the group's list border.

The BorderColor property changes the color for the group's list border. The BorderColor property has no effect if you have not specified the group's list background color using the [BackColorList](#) property.

The following sample shows how to show the border of the group's list:

```
With ExplorerTree1
  With .Groups.Add("Group")
    .BackColorList = vbWhite
    .BorderColor = vbBlack
    .Expanded = True
  End With
End With
```


property Group.Caption as String

Specifies the group's caption.

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

You can use the [Add](#) method to specify the group's caption, when the group is added to groups collection. Use the [CaptionFormat](#) property to allow built-in HTML tags in the group's caption. The Caption property supports the following built-in HTML tags, if the CaptionFormat property is exHTML

- ` ... ` displays the text in **bold**
- `<i> ... </i>` 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.

The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using `<a ;exp=>` or `<a ;e64=>` anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

- exp, stores the plain text to be shown once the user clicks the anchor, such as "`<a ;exp=show lines>`"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "`<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu`" that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY" string encodes the "`<fgcolor 808080>show lines<a>-</fgcolor>`" The `Decode64Text/Encode64Text` methods of the `eXPrint` can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "`<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3`" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show

lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** 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=rrgbb> ... </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=rrgbb> ... </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=rrgbb> ... </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=rrgbb> ... </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
- **number[:width]** 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.
- **key[:width]** 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 **&**; (&), **<**; (<), **>**; (>), **"**; (") 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 FFFFFFFF;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 Group.CaptionFormat as CaptionFormatEnum

Specifies how the group's caption is displayed.

Type	Description
CaptionFormatEnum	A CaptionFormatEnum expression that indicates whether the group's caption uses built-in HTML tags.

By default, the CaptionFormat property is exText. Use the CaptionFormat property to allow built-in HTML tags to the group's caption. Use the [Caption](#) property to access the group's caption.

property Group.CheckImage(State as Long) as Long

Retrieves or sets a value that indicates the image used by cells of checkbox type.

Type	Description
State as Long	A long 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.

Use CheckImage and [RadiolImage](#) properties to define icons for radio and check box cells.

method Group.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.

property Group.ColumnAutoSize as Boolean

Returns or sets a value indicating whether the group will automatically size its visible columns to fit on the group's client width.

Type	Description
Boolean	A boolean expression indicating whether the group will automatically size its visible columns to fit on the group's client width.

Use the ColumnAutoSize property to fit all your columns in the group's client area. By default, the ColumnAutoSize property is True. Use the [HeaderVisible](#) property to show the group's header bar.

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

Retrieves the column 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 X location of the mouse pointer. The x values is always expressed in client coordinates.
Long	A long expression that indicates the column's index, or -1 if there is no column at the point. The property gets a negative value less or equal with 256, if the point is in the area between columns where the user can resize the column.

Use the ColumnFromPoint property to access the column from the point specified by the {X,Y} coordinates. The ColumnFromPoint property gets the index of the column when the cursor hovers the control's header bar. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the ColumnFromPoint property determines the handle of the item from the cursor.** Use the [GroupFromPoint](#) property to get the group's caption from the cursor. Use the [GroupListFromPoint](#) property to get the group's list from cursor. Use the [ItemFromPoint](#) property to get the cell or item from the cursor.

The following VB sample prints the caption of the column over the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim g As EXPLORERTREELibCtl.Group
    With ExplorerTree1
        Set g = .GroupListFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not (g Is Nothing) Then
            With g
                Dim c As Long
                c = .ColumnFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
                If (c >= 0) Then
                    With .Columns(c)
```



```
        Debug.Print .Caption
    End With
End If
End With
End If
End With
End Sub
```

The following VFP sample prints the caption of the column over the cursor:

```
with thisform.Olecontrol1
  local g
  g = .GroupListFromPoint(-1,-1)
  If !isnull(g) Then
    with g
      local c
      c = .ColumnFromPoint(-1,-1)
      If c >= 0 Then
        with .Columns.Item(c)
          wait window .Caption nowait
        endwith
      EndIf
    endwith
  EndIf
endwith
```

property Group.Columns as Columns

Retrieves the group's columns collection.

Type	Description
Columns	A Columns object that holds the group's columns collection.

Use the Columns property to access the group's Columns collection. Use the Columns collection to add, remove or change columns. By default, the group adds a default column. Adding a new item is not allowed if the group has no columns.

property Group.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 Group.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 [CellCaption](#) 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 Group.ContinueColumnScroll as Boolean

Retrieves or sets a value indicating whether the group will automatically scroll the visible columns by pixel or by column width.

Type	Description
Boolean	A boolean expression indicating whether the group will automatically scroll the visible columns by pixel or by column width.

By default, the columns are scrolled pixel by pixel. Use the ContinueColumnScroll to scroll horizontally the group column by column.

property Group.CountLockedColumns as Long

Retrieves or sets a value indicating the number of locked columns. A locked column is not scrollable.

Type	Description
Long	A long expression indicating the number of locked columns.

The group may contains two kind of columns: locked and unlocked. The locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the [CountLockedColumns](#) to specify the number of locked columns. The unlocked contains the columns that can be scrolled horizontally. Use the [BackColorLock](#) property to change the group's background color for the locked area. Use the [LockedItemCount](#) property to lock or unlock items to the top or bottom side of the group. Use the [MergeCells](#) method to combine two or more cells in a single cell.

property Group.DataSource as Object

Retrieves or sets a value that indicates the data source for object.

Type	Description
Object	An ADO.Recordset or DAO object used bind the group to a data source.

The DataSource property uses the ADO/DAO record set objects. Use the [PutItems](#) to load an array to the group. The following sample binds an ADO recordset to an ExplorerTree group:

```
With ExplorerTree1.Groups.Add("Group 1")
  Set rs = CreateObject("ADODB.Recordset")
  rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
  .BeginUpdate
  .ColumnAutoResize = False
  .HeaderVisible = True
  Set .DataSource = rs
  .EndUpdate
End With
```

The DataSource clears the columns collection, and fill the record set into the group, like a list. Use [SetParent](#) method to make your list a hierarchy.

property Group.DefaultItemHeight as Long

Retrieves or sets a value that indicates the default item height.

Type	Description
Long	A long expression indicates the default item height.

Changing the property fails if the group contains already items. You can change the DefaultItemHeight property at design time, or at runtime, before adding any new items to [Items](#) collection.

property Group.Description(Type as DescriptionTypeEnum) as String

Changes descriptions for group objects.

Type	Description
Type as DescriptionTypeEnum	A DescriptionTypeEnum expression that indicates the part being changed.
String	A string value that indicates the part's description.

Use the Description property to customize the captions for group filter bar window. For instance, the `Description(exFilterAll) = "(Include All)"` changes the "(All)" item description in the filter bar window.

property Group.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.

method Group.EndUpdate ()

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

Type

Description

The [BeginUpdate](#) and EndUpdate methods increases the speed of loading your items, by preventing painting the group when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too. The following sample shows how to use the BeginUpdate and EndUpdate methods:

```
With ExplorerTree1.Groups.Add("Group 1")
  Set rs = CreateObject("ADODB.Recordset")
  rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
  .BeginUpdate
  .ColumnAutoResize = False
  .HeaderVisible = True
  ' By default, the group adds a default colum, so we remove it first
  .Columns.Clear
  For Each f In rs.Fields
    .Columns.Add f.Name
  Next
  .PutItems rs.GetRows()
  .EndUpdate
End With
```

method Group.EnsureVisibleColumn (Column as Variant)

Scrolls the group's content to ensure that the column fits the group's client area.

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

The `EnsureVisibleColumn` method ensures that the given column fits the group's client area. The `EnsureVisibleColumn` method has no effect if the column is hidden. Use the [Visible](#) property to show or hide a column. Use the [Position](#) property to change the column's position. Use the [EnsureVisibleItem](#) method to ensure that an item fits the group's client area.

property Group.Expanded as Boolean

Expands or collapses the group.

Type	Description
Boolean	A boolean expression that indicates whether the group is expanded or collapsed.

Use the Expanded property to expand or collapse the group. The control fires [BeforeExpandItem](#) event before expanding or collapsing the group. The [AfterExpandGroup](#) event is fired after user expands a group. The [Height](#) property specify the height of the group's list when it is expanded.

property Group.ExpandOnDbfClick 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 ExpandOnDbfClick property to disable expanding or collapsing items when user dbl clicks an item. By default, the ExpandOnDbfClick property is True. 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.

property Group.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 [ExpandOnDbClick](#) property to specify whether the control expands or collapses a node when user db 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 expand or collapse the items using the + or - keys on the numeric keypad.

property Group.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. 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.

property `Group.FilterBarBackColor` as `Color`

Specifies the background color of the group's filter bar.

Type	Description
Color	A color expression that defines the background color for description of the group's filter.

Use the [FilterBarForeColor](#) and `FilterBarBackColor` properties to define the colors used to paint the description for group's filter bar.

property Group.FilterBarCaption as String

Specifies the filter bar's caption.

Type	Description
String	A string expression that indicates the caption in the filter bar.

Use the FilterBarCaption property to change the caption of the group's filter bar.

property Group.FilterBarDropDownHeight as Double

Specifies the height of the drop down filter window proportionally with the height of the group'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. Use the [DisplayFilterButton](#) property to display a filter button to the column's header. 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 group's list.

If the FilterBarDropDownHeight property is negative, the absolute value of the FilterBarDropDownHeight property indicates the height of the drop down filter window in pixels. In this case, the height of the drop down filter window is not proportionally with the height of the group's list area. For instance, the following sample specifies the height of the drop down filter window being 100 pixels:

```
With Group
    .FilterBarDropDownHeight = -100
End With
```

If the FilterBarDropDownHeight property is greater than 0, it indicates the height of the drop down filter window proportionally with the height of the group's height list. For instance, the following sample specifies the height of the drop down filter window being the same with the height of the group's list area:

```
With Group
    .FilterBarDropDownHeight = 1
End With
```

The drop down filter window always include an item.

Group 1		
OrderID	Custo...	Employe...
10330	RATTC	3
10331	VINET	9
10332	TOMSP	3
10333	HANAR	5
10334	SUPRD	8
10335	CHOPS	7
10336	FRANK	7
10337	FRANK	4
10338	OLDWO	4

property **Group.FilterBarFont** as **IFontDisp**

Retrieves or sets the font for group's filter bar.

Type	Description
IFontDisp	A font object that indicates the font used to paint the description for group's filter

Use the `FilterBarFont` property to specify the font for the group's filter bar.

property Group.FilterBarForeColor as Color

Specifies the foreground color of the group's filter bar.

Type	Description
Color	A color expression that defines the foreground color of the description of the group's filter.

Use the FilterBarForeColor and [FilterBarBackColor](#) properties to define colors used to paint the description of the group's filter bar.

property Group.FilterBarHeight as Long

Specifies the height of the group's filter bar. If the value is less than 0, the filter bar is automatically resized to fit its description.

Type	Description
Long	A long expression that indicates the height of the filter bar status.

The filter bar status defines the group's filter description. If the FilterBarHeight property is less than 0 the group automatically updates the height of the filter's description to fit in the group's client area. If the FilterBarHeight property is zero the filter's description is hidden. If the FilterBarHeight property is greater than zero it defines the height in pixels of the filter's description. Use the [ClearFilter](#) method to clear the group's filter.

property Group.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 "`<i><fgcolor=808080>Start Filter...</fgcolor></i>`". 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. The control fires the [FilterChanging](#) event just before applying the filter, and [FilterChange](#) once the list gets filtered. Use the [FilterBarCaption](#) property to change the caption in the filter bar once a new filter is applied. The [FilterBarFont](#) property specifies the font to be used in the filter bar. The [FilterBarBackColor](#) property specifies the background color or the visual aspect of the control's filter bar. The [FilterBarForeColor](#) property specifies the foreground color or the control's filter bar.

The FilterBarPrompt property supports HTML format as described here:

- ` ... ` displays the text in **bold**
- `<i> ... </i>` 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.

The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using `<a ;exp=>` or `<a ;e64=>` anchor tags. The `exp/e64` field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.



- `exp`, stores the plain text to be shown once the user clicks the anchor, such as "`<a ;exp=show lines>`"
- `e64`, encodes in BASE64 the HTML text to be shown once the user clicks the

anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** 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=rrggb> ... </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=rrggb> ... </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=rrggb> ... </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=rrggb> ... </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
- **number[:width]** 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.

- **key[:width]** 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 FFFFFFFF;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:



The FilterBarPrompt property has effect only if:

- [FilterBarPromptVisible](#) property is True
- [FilterBarPromptPattern](#) property is Empty.

property Group.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 Group.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 pattern 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 Group.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. The control fires the [FilterChanging](#) event just before applying the filter, and [FilterChange](#) once the list gets filtered. 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 Group.FilterBarPromptVisible as FilterBarVisibleEnum


Shows or hides the filter prompt.

Type	Description
FilterBarVisibleEnum	A FilterBarVisibleEnum expression that specifies whether the filter prompt field is visible or hidden. The filter prompt is shown in the bottom part of the control, where the filter bar is shown.

By default, The FilterBarPromptVisible property is exFilterBarHidden(0). 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 [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. The control fires the [FilterChanging](#) event just before applying the filter, and [FilterChange](#) once the list gets filtered. Use the [FilterBarCaption](#) property to change the caption in the filter bar once a new filter is applied.

The following screen show shows the filter prompt (FilterBarPromptVisible property is True):

Name	Title	City
Nancy Davolio	Sales Representative	Seattle
Andrew Fuller	Vice President, Sales	Tacoma
Janet Leverling	Sales Representative	Kirkland
Margaret Peacock	Sales Representative	Redmond
Steven Buchanan	Sales Manager	London
Michael Suyama	Sales Representative	London
Robert King	Sales Representative	London
Laura Callahan	Inside Sales Coordinator	Seattle
Anne Dodsworth	Sales Representative	London



The following screen show shows the list once the user types "london":

Name	Title	City
Steven Buchanan	Sales Manager	London
Michael Suyama	Sales Representative	London
Robert King	Sales Representative	London
Anne Dodsworth	Sales Representative	London

 london|

property Group.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 treeed 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 treeed 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.

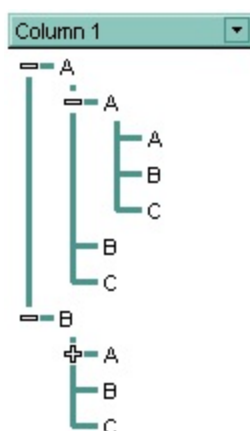
property Group.FilterInclude as FilterIncludeEnum

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.

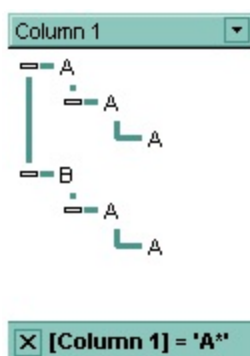
By default, the FilterInclude property is `exItemsWithoutChlds`. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the [Filter](#) property and [FilterType](#) property to specify the column's filter. Use the [ApplyFilter](#) to apply the filter at runtime. Use the [ClearFilter](#) method to clear the control's filter.

Let's say that we have the following hierarchy:

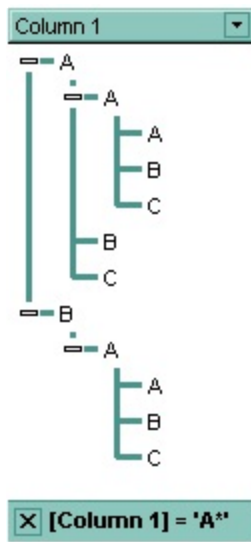


and the [Filter](#) property is "A*", [FilterType](#) property is FilterPattern.

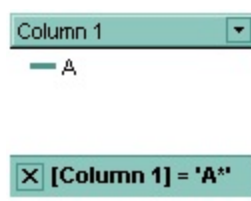
If the FilterInclude property is `exItemsWithoutChlds`, the filtered list looks like follows:



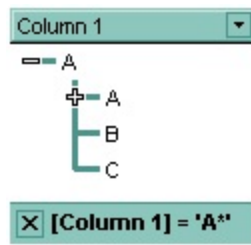
If the FilterInclude property is `exItemsWithChlds`, the filtered list looks like follows:



If the FilterInclude property is **exRootsWithoutChilds**, the filtered list looks like follows:



If the FilterInclude property is **exRootsWithChilds**, the filtered list looks like follows:



property Group.Font as IFontDisp

Retrieves or sets the group's font.

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

Use the Font property to change the group's font . Use the [Font](#) property to change the control's font.

property Group.ForeColor as Color

Specifies the group's foreground color.

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

Use the ForeColor property to specify the group's caption foreground color. Use the [ForeColorGroup](#) property to specify the default foreground color. Use the [ForeColorList](#) property to specify the foreground color of the group's list.

property Group.ForeColorHeader as Color

Specifies the header's foreground color.

Type	Description
Color	A color expression that indicates the foreground color for group's header.

Use the [BackColorHeader](#) and ForeColorHeader properties to customize the group's header bar.

property Group.ForeColorList as Color

Retrieves or sets a value that indicates the foreground color of the group's list when it is active.

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

Use the ForeColorList property to specify the foreground color for the group's list. Use the [ForeColor](#) property to specify the foreground color of the group's caption. If you have not specified a foreground color for the group's list the control's [ForeColor](#) property specify the foreground color of the group's list.

property Group.ForeColorLock as Color

Retrieves or sets a value that indicates the group's foreground color for the locked area.

Type	Description
Color	A color expression that indicates the group's foreground color for the locked area.

The ExplorerTree group can group the group columns into two categories: locked and unlocked. The Locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the [CountLockedColumns](#) to specify the number of locked columns. The unlocked are contains the columns that can be scrolled horizontally. To change the background color of the group's locked area use [BackColorLock](#) property.

property Group.FullRowSelect as Boolean

Enables full-row selection in the group.

Type	Description
Boolean	A boolean expression that indicates whether the group support full-row selection.

For instance, you can set the FullRowSelect to False, when your list acts like a simple tree control. The column pointed by the [SelectColumnIndex](#) specifies the column where the selected cell is marked.

method Group.GetItem (Options as Variant)

Gets the collection of items into a safe array,

Type	Description
Options as Variant	<p>Specifies a long expression as follows:</p> <ul style="list-style-type: none">• if 0, the result is a two-dimensional array with cell's captions. The list includes the <i>collapsed</i> items, and the items are included as they are displayed (sorted, filtered). This option exports the captions of cells. This option exports the captions of the cells (CellCaption property)• if 1, the result the one-dimensional array of handles of items in the control as they are displayed (sorted, filtered). The list <i>does not include the collapsed items</i>. 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 <i>collapsed</i> items, and the items are included as they are displayed (sorted, filtered). This option exports the captions of the cells (CellCaption 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	<p>A safe array that holds the items in the control. If the control has a single column, the GetItem returns a single dimension array (object[]), else The safe array being returned has two dimensions (object[,]). The first</p>

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. If the `Options` parameter is 0, the `GetItems` method collect the child items as well, no matter if the parent item is collapsed or expanded. Use the [PutItems](#) method to load an array to the group. The method returns nothing if the group has no columns or items. Use the [Items](#) property to access the items collection. 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.

/NET Assembly:

The following C# sample converts the returned value to a `object[]` when the control contains a single column:

```
object[] Items = (object[])group.GetItems()
```

or when the control contains multiple columns, the syntax is as follows:

```
object[,] Items = (object[,])group.GetItems()
```

The following VB.NET sample converts the returned value to a `Object()` when the control contains a single column:

```
Dim Items As Object() = group.GetItems()
```

or when the control contains multiple columns, the syntax is as follows:

```
Dim Items As Object(,) = group.GetItems()
```

/COM version:

The following sample gets the items from a group and put them to the second one:

```
With Group2
    .BeginUpdate
    .Columns.Clear
    Dim c As EXPLORERTREELibCtl.Column
    For Each c In Group1.Columns
        .Columns.Add c.Caption
    Next
```

.PutItems Group1.GetItems

.EndUpdate

End With

property Group.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.

property Group.GridLineStyle as GridLineStyleEnum

Specifies the style for gridlines in the list part of the control.

Type	Description
GridLineStyleEnum	A GridLineStyleEnum expression that specifies the style to show the control's horizontal or vertical lines.

By default, the GridLineStyle property is exGridLinesDot. The GridLineStyle property has effect only if the [DrawGridLines](#) property is not zero. The GridLineStyle property can be used to specify the style for horizontal or/and vertical grid lines. Use the [GridLineColor](#) property to specify the color for grid lines. Use the [LinesAtRoot](#) property specifies whether the control links the root items of the control. Use the [HasLines](#) property to specify whether the control draws the link between child items to their corresponding parent item.

The following VB sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = GridLineStyleEnum.exGridLinesHDash Or  
GridLineStyleEnum.exGridLinesVSolid
```

The following VB/NET sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = exontrol.EXGRIDLib.GridLineStyleEnum.exGridLinesHDash Or  
exontrol.EXGRIDLib.GridLineStyleEnum.exGridLinesVSolid
```

The following C# sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = exontrol.EXGRIDLib.GridLineStyleEnum.exGridLinesHDash |  
exontrol.EXGRIDLib.GridLineStyleEnum.exGridLinesVSolid;
```

The following Delphi sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle := Integer(EXGRIDLib.GridLineStyleEnum.exGridLinesHDash) Or  
Integer(EXGRIDLib.GridLineStyleEnum.exGridLinesVSolid);
```

The following VFP sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = 36
```


property Group.HasButtons as ExpandButtonEnum

Adds a button to the left side of each parent item.

Type	Description
ExpandButtonEnum	An ExpandButtonEnum expression that indicates whether the left side button of each parent item is visible or hidden.

The HasButtons property has effect only if the data is displayed as a tree. Use the [InsertItem](#) property to insert child items. The user can click the button to expand or collapse the child items as an alternative to double-clicking the parent item. Use [ExpandItem](#) property of Items to programmatically expand/collapse items. Use the [HasButtonsCaption](#) property to assign custom icons for +/- buttons.

property Group.HasButtonsCustom(Expanded as Boolean) as Long

Specifies the index of icons for +/- signs when the HasButtons property is exCustom.

Type	Description
Expanded as Boolean	A boolean expression that indicates the sign being changed.
Long	A long expression that indicates the icon being used for +/- signs on the parent items.

Use the HasButtonsCustom property to assign custom icons to the +/- signs on the parent items. The HasButtonsCustom property has effect only if the [HasButtons](#) property is exCustom. Use the [Images](#), [Replacelcon](#) methods to add new icons to the control.

The following sample assigns different icons for +/- buttons:

```
With ExplorerTree1
```

```
  .BeginUpdate
```

```
  .BackColor = vbWhite
```

```
  .ForeColor = vbBlack
```

```
  .BackColorGroup2 = SystemColorConstants.vb3DShadow
```

```
  .Images
```

```
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
```

```
  .Images
```

```
"gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjlBAEijUlK8pIUrlktl0vmExn
```

```
With .Groups.Add("Group")
```

```
  .BeginUpdate
```

```
    .Expanded = True
```

```
    .FullRowSelect = False
```

```
    .HasButtons = exCustom
```

```
    .HasButtonsCustom(False) = 1
```

```
    .HasButtonsCustom(True) = 2
```

```
    .LinesAtRoot = exLinesAtRoot
```

```
  Dim h As HITEM
```

With .Items

```
h = .AddItem("Item 1")
```

```
.CellImage(h, 0) = 3
```

```
.CellCaptionFormat(h, 0) = exHTML
```

```
.InsertItem .InsertItem(h, , "SubItem 1"), , "SubItem 1.1"
```

```
.InsertItem h, , "SubItem 2"
```

End With

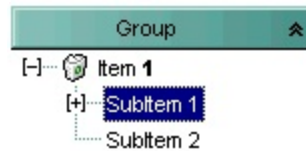
```
.EndUpdate
```

End With

```
.EndUpdate
```

End With

Running the sample you get:



property Group.HasLines as HierarchyLineEnum

Enhances the graphic representation of a tree group's hierarchy by drawing lines that link child items to their corresponding parent item.

Type	Description
HierarchyLineEnum	An HierarchyLinesEnum expression that indicates whether the group uses the lines to link the items of the hierarchy.

Use the HasLines property to hide the hierarchy lines. Use the [LinesAtRoot](#) property to allow group displays a line that links that root items of the group. Use the [InsertItem](#) method to insert child items to the group. Use [HasButtons](#) property to hide the buttons displayed at the left of each parent item. Use the [DrawGridLines](#) property to display grid lines.

property Group.HeaderAppearance as AppearanceEnum

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

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the header's appearance.

Use the HeaderAppearance property to change the appearance of the group's header bar. Use the [HeaderVisible](#) property to hide the group's header bar.

property Group.HeaderHeight as Long

Retrieves or sets a value indicating the group's header height.

Type	Description
Long	A long expression that indicates the height of the group's header bar.

Use the `HeaderHeight` property to change the height of the group's header bar. Use the [HeaderVisible](#) property to show the group's header bar. *If the [HeaderSingleLine](#) property is False, the `HeaderHeight` property specifies the maximum height of the control's header when the user resizes the columns.*

For instance, the following sample displays the group's header bar using multiple lines:

```
With ExplorerTree1.Groups.Add("Group 2")
    .BeginUpdate
        .HeaderVisible = True
        .HeaderHeight = 32
        With .Columns(0)
            .HTMLCaption = "Line1 <br> Line2"
            .Width = 128
        End With
        With .Columns.Add("Column 1")
            .HTMLCaption = "Line1 <br> Line2"
            .Width = 128
        End With
    .EndUpdate
End With
```

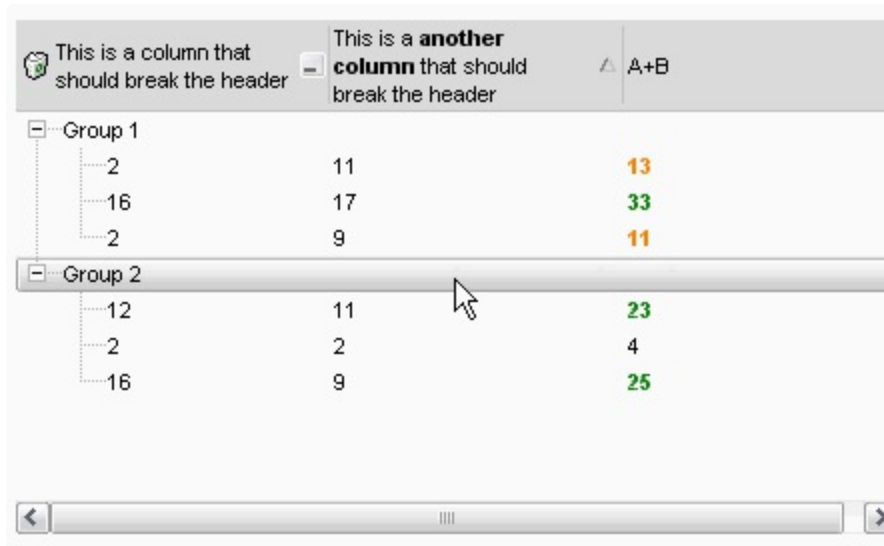
property Group.HeaderSingleLine as Boolean

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 default, 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.

The following screen show shows the control's header while it displays a multiple lines (HeaderSingleLine = False):



The screenshot shows a data grid control with a header and two data groups. The header has three columns: 'This is a column that should break the header', 'This is a **another column** that should break the header', and 'A+B'. The data is organized into two groups, 'Group 1' and 'Group 2'. The first column of data is broken into multiple lines. The second column has a bolded 'another' in its header. The third column contains the sum of the first two columns.

This is a column that should break the header	This is a another column that should break the header	A+B
Group 1		
2	11	13
16	17	33
2	9	11
Group 2		
12	11	23
2	2	4
16	9	25

The following screen shot shows the control's header on multiple levels using the [LevelKey](#) property:

Level 1		
Level 2		
This is a colu...	This is a another colu...	A+B
Level 3		
Group 1		
2	11	13
16	17	33
2	9	11
Group 2		
12	11	23
2	2	4
16	9	25

The following screen show shows the control's header while it displays a single line (`HeaderSingleLine = True`):

This is a column that s...			This is a another column ..	A+B
Group 1				
2	11		13	
16	17		33	
2	9		11	
Group 2				
12	11	↔	23	
2	2		4	
16	9		25	

property Group.HeaderVisible as Boolean

Retrieves or sets a value that indicates whether the the group's header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the the group's header is visible or hidden.

By default, the group's header bar is hidden. Use the HeaderVisible property to show the group's header bar. Use the [ColumnAutoResize](#) property to fit all your columns in the group's client area. Use the [HeaderAppearance](#) property to change the header bar's appearance.

property Group.Height as Long

Retrieves the height in pixels of the group's list.

Type	Description
Long	A long expression that indicates the height in pixels of the group's list.

The `Left`, `Top`, `Width` and `Height` properties determine the location of the group's list in client coordinates. The [AutoHeight](#) property specifies whether the `Height` property is computed based on the items in the group's list. The [BorderWidth](#) and [BorderHeight](#) properties specify the size of the control's border.

Use the [LayoutChanged](#) event and [Left](#), [Top](#), [Width](#) and `Height` to determine the coordinates of the group's list on the fly.

property Group.HideSelection as Boolean

Returns a value that determines whether selected item appears highlighted when a group loses the focus.

Type	Description
Boolean	A boolean expression that indicates whether the selected item appears highlighted when a group loses the focus.

By default, the HideSelection property is False. You can use this property to indicate which item is highlighted while another form or a dialog box has the focus.

property Group.HotBackColor as Color

Retrieves or sets a value that indicates the hot-tracking background color.

Type	Description
Color	A color expression that specifies the hot-tracking background color

By default, the HotBackColor property is 0, which indicates no effect. The HotBackColor property specifies hot-tracking background color. The [HotForeColor](#) property defines the hot-tracking foreground color.

property Group.HotForeColor as Color

Retrieves or sets a value that indicates the hot-tracking foreground color.

Type	Description
Color	A color expression that defines the hot-tracking foreground color.

By default, the HotForeColor property is 0, which indicates no effect. The HotForeColor property defines the hot-tracking foreground color. The [HotBackColor](#) property specifies hot-tracking background color.

property Group.hWnd as Long

Retrieves the group's window handle.

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

Use the hWnd property to get the handle of the group's list window. Use the [ItemWindowHost](#) property to get the handle of the container window that host an item's ActiveX Control. Use the [hWnd](#) property to get the handle of the control's window.

property Group.HyperLinkColor as Color

Specifies the hyperlink color.

Type	Description
Color	A color expression that specifies the hyperlink color.

Use the HyperLinkColor property to specify the color used when the cursor is over the hyperlink cells. A hyperlink cell has the [CellHyperLink](#) property true.

property Group.Image as Variant

Specifies the index of the group's icon.

Type	Description
Variant	A long expression that indicates the index of icon being used, a string expression that indicates the base64 encoded string that holds a picture object, a Picture object. Use the eximages tool to save your picture as base64 encoded format.

Use the Image property to assign an icon or a picture to the group. Use the [ImageAlignment](#) property to align the image inside the group's header. Use the [Images](#) and [Replacelcon](#) methods to update the images list collection, at runtime. Use the [HTMLPicture](#) property to add custom-sized pictures to your caption.

The following sample adds a group that displays a custom size picture using BASE 64 encoding:

```
With ExplorerTree1.Groups
  With .Add("new")
    Dim s As String
    s =
"gbHJJGHA5MIqAAXAD3AENhozhpmhqZhrMhr/h0QGcQM0QTMQZkQf8QAESGcSM0STM

    s = s +
"JE6QQCj2UBhE0UAHGscgUEmlZXGqVQ1kclg/CYcwllEToBGiZwlHoPAYkEAYwBWHAUHGAB,

    .Image = s
  End With
End With
```


property Group.ImageAlignment as AlignmentEnum

Specifies the icon's alignment.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the image in the group's caption.

By default, the ImageAlignment property is exLeft. Use the ImageAlignment property to align the image inside the group's caption. Use the [Image](#) property to assign an icon to the group's caption. The ImageAlignment property has no effect if the group's caption displays no image.

property Group.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.

Use `Indent = 0` to ignore the amount, in pixels, that child items are indented relative to their parent items. Use the [IndentGroupLeft](#) and [IndentGroupRight](#) properties to indent the group's list to the left or to the right side. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bott

property Group.IndentGroupLeft as Long

Specifies a value that indicates the indent of the group's list to the left side.

Type	Description
Long	A long expression that indicates the indent of the group's list to the left side in pixels.

The IndentGroupLeft property is 0. Use the [IndentGroupRight](#) property to specify the indent of the group's list to the right side. The group's list client area is computed based on the [BorderWidth](#), [BorderHeight](#), IndentGroupLeft, IndentGroupRight and [BorderGroupHeight](#) properties. Use the [Left](#), [Top](#), [Width](#) and [Height](#) to determine the client coordinates of the group's list. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bottom side.

property Group.IndentGroupRight as Long

Specifies a value that indicates the indent of the group's list to the right side.

Type	Description
Long	A long expression that indicates the indent of the group's list to the right side in pixels.

The IndentGroupRight property is 0. Use the [IndentGroupLeft](#) property to specify the indent of the group's list to the left side. The group's list client area is computed based on the [BorderWidth](#), [BorderHeight](#), IndentGroupLeft, IndentGroupRight and [BorderGroupHeight](#) properties. Use the [Left](#), [Top](#), [Width](#) and [Height](#) to determine the client coordinates of the group's list. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bott

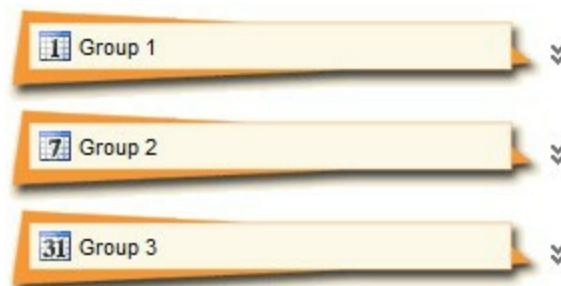
property Group.IndentHeaderBottom as Long

Specifies the number of pixels to indent the group's header from the bottom part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the bottom side.

By default, the `IndentHeaderTop` property is 0. The `IndentHeaderLeft` property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the `IndentHeaderBottom` property to indent the group's header from bottom side. Use the [Indent](#) property to indent the child items being displayed in the group's list. Use the [IndentGroupLeft](#) and [IndentGroupRight](#) properties to specify the left and right indentation of the group's list.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



property Group.IndentHeaderLeft as Long

Specifies the number of pixels to indent the group's header from the left part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the left side.

By default, the `IndentHeaderLeft` property is 0. The `IndentHeaderLeft` property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the `IndentHeaderLeft` property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bottom side. Use the [Indent](#) property to indent the child items being displayed in the group's list. Use the [IndentGroupLeft](#) and [IndentGroupRight](#) properties to specify the left and right indentation of the group's list.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



The following VFP sample changes the visual appearance for the groups:

```
with thisform.ExplorerTree1
```

```

.BeginUpdate
with VisualAppearance
  .Add(1,"c:\images\group.ebn")
  .Add(2,"CP:1 0 0 -20 0")
endwith
.BackColorGroup = 0x2000000
.GroupHeight = 48
with .Groups
  with .Add("Group 1")
    .Alignment = 0
    .IndentHeaderLeft = 12
    .IndentHeaderTop = -8
  endwith
  with .Add("Group 2")
    .Alignment = 0
    .IndentHeaderLeft = 12
    .IndentHeaderTop = -8
  endwith
  with .Add("Group 3")
    .Alignment = 0
    .IndentHeaderLeft = 12
    .IndentHeaderTop = -8
  endwith
endwith
.EndUpdate
endwith

```

The following Delphi sample changes the visual appearance for the groups:

```

with AxExplorerTree1 do
begin
  BeginUpdate();
  with VisualAppearance do
  begin
    Add(1,'c:\images\group.ebn');
    Add(2,'CP:1 0 0 -20 0');
  end;
end;

```

```

(GetOcx() as EXPLORERTREELib.ExplorerTree).BackColorGroup := $2000000;
GroupHeight := 48;
with Groups do
begin
  with Add('Group 1') do
  begin
    Alignment := EXPLORERTREELib.AlignmentEnum.LeftAlignment;
    IndentHeaderLeft := 12;
    IndentHeaderTop := -8;
  end;
  with Add('Group 2') do
  begin
    Alignment := EXPLORERTREELib.AlignmentEnum.LeftAlignment;
    IndentHeaderLeft := 12;
    IndentHeaderTop := -8;
  end;
  with Add('Group 3') do
  begin
    Alignment := EXPLORERTREELib.AlignmentEnum.LeftAlignment;
    IndentHeaderLeft := 12;
    IndentHeaderTop := -8;
  end;
end;
EndUpdate();
end

```

Here's the Template that generates the screens:

```

BeginUpdate
VisualAppearance
{
  Add(1,"c:/images/group.ebn")
  Add(2, "CP:1 0 0 -20 0")
}

```

```

BackColorGroup = 33554432
GroupHeight = 48

```


Groups

```
{  
  "Group 1"  
  {  
    Alignment = 0  
    IndentHeaderLeft = 12  
    IndentHeaderTop = -8  
  }  
  "Group 2"  
  {  
    Alignment = 0  
    IndentHeaderLeft = 12  
    IndentHeaderTop = -8  
  }  
  "Group 3"  
  {  
    Alignment = 0  
    IndentHeaderLeft = 12  
    IndentHeaderTop = -8  
  }  
}  
EndUpdate
```

You can find the EBN being used [here](#).

property Group.IndentHeaderRight as Long

Specifies the number of pixels to indent the group's header from the right part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the right side.

By default, the IndentHeaderRight property is 0. The IndentHeaderLeft property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the IndentHeaderRight property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bottom side. Use the [Indent](#) property to indent the child items being displayed in the group's list. Use the [IndentGroupLeft](#) and [IndentGroupRight](#) properties to specify the left and right indentation of the group's list.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



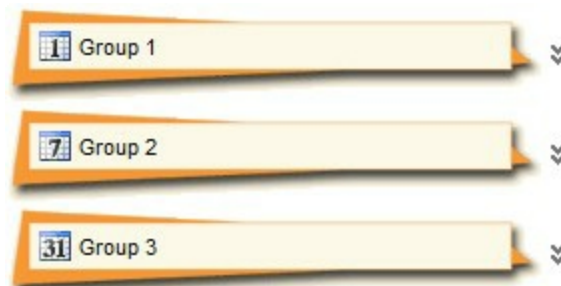
property Group.IndentHeaderTop as Long

Specifies the number of pixels to indent the group's header from the top part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the top side.

By default, the IndentHeaderTop property is 0. The IndentHeaderLeft property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the IndentHeaderTop property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bottom side. Use the [Indent](#) property to indent the child items being displayed in the group's list. Use the [IndentGroupLeft](#) and [IndentGroupRight](#) properties to specify the left and right indentation of the group's list.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



property Group.Index as Long

Retrieves the index of the object into the Groups collection..

Type	Description
Long	A long expression that indicates the group's index into the groups collection.

Use the Index property to identify a Group object into the groups collection. Use the [Position](#) property to specify the group's position.

property Group.Italic as Boolean

Specifies a value that indicates whether the group's caption should appear in italic.

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

Use the [Bold](#), [Italic](#), [Underline](#) and [StrikeOut](#) properties to customize the caption's font attributes.

property Group.ItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, ColIndex as Long, HitTestInfo as HitTestInfoEnum) as HITEM

Retrieves the item 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.
ColIndex as Long	A long expression that indicates on return, the column where the point belongs. If the return value is zero, the ColIndex may indicate the handle of the cell (inner cell).
HitTestInfo as HitTestInfoEnum	A HitTestInfoEnum expression that determines on return the position of the cursor within the cell.
HITEM	A long expression that indicates the item's handle where the point is.

Use the ItemFromPoint property to get the item from the point specified by the {X,Y}. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the ItemFromPoint property determines the handle of the item from the cursor.** Use the [ColumnFromPoint](#) property to access the column over the point. Use the [GroupFromPoint](#) property to get the group's caption from the cursor. Use the [GroupListFromPoint](#) property to get the group's list from cursor.

The following VB sample prints the cell's caption from the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ExplorerTree1
        ' Determines the group's list from cursor
        Dim g As EXPLORERTREELibCtl.Group
        Set g = .GroupListFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not g Is Nothing Then
            ' Determines the item from the cursor within the group
            Dim h As HITEM, c As Long, hit As EXPLORERTREELibCtl.HitTestInfoEnum
```

```

h = g.ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, hit)
If Not (h = 0) or Not ( c = 0 ) Then
    Debug.Print g.Items.CellCaption(h, c) & " HT = " & hit
End If
End If
End With
End Sub

```

The following VFP sample prints the cell's caption from the cursor:

```

*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform.Olecontrol1
    local g
    g = .GroupListFromPoint(-1,-1)
    If !isnull(g) Then
        with g
            local h, c, hit
            c = 0
            hit = 0
            h = .ItemFromPoint(-1,-1,@c,@hit)
            If h # 0 Then
                with .Items
                    .DefaultItem = h
                    wait window .CellCaption(0, c) nowait
                endwhile
            EndIf
        endwhile
    EndIf
endwith

```

property Group.Items as Items

Retrieves the group's item collection.

Type	Description
Items	An Items object that holds the group's items collection.

Use the Items property to access the group's Items collection. Use the Items collection to add, remove or change the group items. Use the [GetItems](#) method to get items collection into a safe array. Use the [PutItems](#) method to load items from a safe array.

The following VBA sample enumerates all groups and items being shown on the control:

```
Dim e As EXPLOREERTREELib.ExplorerTree
Set e = ExplorerTree1.Object
With e
    Dim g As EXPLOREERTREELib.Group
    For Each g In .Groups
        Debug.Print "Enumerate Group " & g.Caption
        For Each i In g.Items
            With g.Items
                Debug.Print "Item " & .CellCaption(i, 0)
            End With
        Next
    Next
Next
End With
```

The following VBA sample enumerates all groups and items being checked on the control:

```
Dim e As EXPLOREERTREELib.ExplorerTree
Set e = ExplorerTree1.Object
With e
    Dim g As EXPLOREERTREELib.Group
    For Each g In .Groups
        Debug.Print "Enumerate Group " & g.Caption
        For Each i In g.Items
            With g.Items
                If Not (.CellState(i, 0) = 0) Then
                    Debug.Print "Item " & .CellCaption(i, 0)
                End If
            End With
        Next
    Next
Next
End With
```


End If

End With

Next

Next

End With

property Group.ItemsAllowSizing as ItemsAllowSizingEnum

Retrieves or sets a value that indicates whether a user can resize items at run-time.

Type	Description
ItemsAllowSizingEnum	A ItemsAllowSizingEnum expression that indicates whether the user can resize the items at run-time.

By default, the ItemsAllowSizing property is exNoSizing(0). 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. The DefaultItemHeight property affects only items that are going to be added. It doesn't affect items already added.

property Group.Left as Long

Specifies the distance between the left edge of the control and group's list.

Type	Description
Long	A long expression that specifies the distance between the left edge of the control and group's list

The Left, Top, Width and Height properties determines the location of the group's list in client coordinates.

Use the [LayoutChanged](#) event and Left, [Top](#), [Width](#) and [Height](#) to determine the coordinates of the group's list on the fly.

property Group.LinesAtRoot as LinesAtRootEnum

Link items at the root of the hierarchy.

Type	Description
LinesAtRootEnum	A LinesAtRootEnum expression that indicates whether the group link items at the root of the hierarchy.

The group 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 group's hierarchy by drawing lines that link child items to their corresponding parent item.

property Group.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.

Use [SearchColumnIndex](#) property to change the current searching column. Use the [AutoSearch](#) property to enable or disable the incremental searching feature

property Group.Picture as IPictureDisp

Retrieves or sets a graphic to be displayed in the group.

Type	Description
IPictureDisp	A Picture object that's displayed on the group's background.

By default, the group has no picture associated. The group uses the [PictureDisplay](#) property to determine how the picture is displayed on the group's background.

property Group.PictureDisplay as PictureDisplayEnum

Retrieves or sets a value that indicates the way how the graphic is displayed on the group'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 group's background. If the group has no picture associated the PictureDisplay property has no effect.

property Group.PictureDisplayLevelHeader as PictureDisplayEnum

Retrieves or sets a value that indicates the way how the graphic is displayed on the control's header background.

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way how the picture is displayed on the control's header.

Use the PictureDisplayLevelHeader property to arrange the picture on the control's multiple levels header bar. Use the [PictureLevelHeader](#) property to load a picture on the control's header bar when it displays multiple levels. The control's header bar displays multiple levels if there are two or more neighbor columns with the same non empty level key. Use the [LevelKey](#) property to specify the control's level key.

property Group.PictureLevelHeader as IPictureDisp

Retrieves or sets a graphic to be displayed in the control's header when multiple levels is on.

Type	Description
IPictureDisp	A Picture object being displayed on the control's header bar when multiple levels is on.

Use the `PictureLevelHeader` property to display a picture on the control's header bar when it displays the columns using multiple levels. Use the [PictureDisplayLevelHeader](#) property to arrange the picture on the control's multiple levels header bar. The control's header bar displays multiple levels if there are two or more neighbor columns with the same non empty level key. Use the [LevelKey](#) property to specify the control's level key. Use the [Picture](#) property to display a picture on the control's list area. Use the [BackColorLevelHeader](#) property to specify the background color for parts of the control's header bar that are not occupied by column's headers.



property Group.Position as Long

Specifies the group's position.

Type	Description
Long	A long expression that indicates the group's position.

Use the Position property to arrange groups. Use the [Index](#) property to identify a group. Use the [ItemByPos](#) property to access groups by position. The Position property is zero based.

method Group.PutItems (Items as Variant, [Parent as Variant])

Adds data to the control from a SafeArray containing numbers, strings, dates, or nested SafeArrays of numbers, strings, and dates, positioning them as child items of the specified parent item

Type

Description

An array that control uses to fill with. The array can be one or two- dimensional. If the array is one-dimensional, the control requires one column being added before calling the PutItems method. If the Items parameter indicates a two-dimensional array, the first dimension defines the columns, while the second defines the number of items to be loaded. For instance, a(2,100) means 2 columns and 100 items.

For instance:

- `PutItems Array("Item 1", "Item 2", "Item 3")`, adds the rows at the end of the list
- `PutItems Array("Root", Array("Child 1", "Child 2"))`, adds data in a hierarchical structure, at the end of the list
- `PutItems rs.GetRows()`, appends data from a recordset using the GetRows method of the Recordset
- `PutItems rs.GetRows(10)`, inserts the first 10 records from a Recordset using the GetRows method, at the end of the list

Items as Variant

where GetRows() method in ADO retrieves multiple records from a Recordset object and stores them in a two-dimensional array.

Indicates one of the following:

- missing, `empty` or `0` {number}, specifies that the data(Items) is being appended (added to the end of the list)
- a `long` expression, that specifies the handle of the item where the array is being inserted
- a string expression of of `"parent;IDColumn;ParentIDColumn"` format, where,

Parent as Variant

'parent' denotes the handle of the item where the data is being inserted, 'IDColumn' refers to the index of the column containing row identifiers, and 'ParentIDColumn' indicates the index of the column containing identifiers of parent rows. This way, you can insert data hierarchically using parent-id relationship. A parent-id relationship is a way of organizing data in a hierarchical structure where each element (or "child") is associated with a parent element. Please be aware that the rows of the data are inserted as they were provided by the Items parameter. Therefore, it is important that the data provided be sorted by the IDColumn so that the parent row referred to by the ParentIDColumn value is already present and can be used to insert the current row as a child of it.

For instance:

- `PutItems Array("Item 1", "Item 2", "Item 3"), Items.ItemByIndex(2)`, inserts the rows as children of the item with index 2
- `PutItems Array("Root", Array("Child 1", "Child 2")), Items.FirstVisibleItem`, Inserts data as a hierarchical structure, placing it as a child of the first visible item
- `PutItems rs.GetRows(), Items.ItemByIndex(0)`, inserts the records from the recordset using the GetRows method of the Recordset, placing them as children of the item with index 0
- `PutItems rs.GetRows(), ";0;3"`, inserts the records from the recordset using the GetRows method of the Recordset, utilizing parent-child relationships. The first column (index 0) contains the identifiers of the rows, while the fourth column (index 3) contains the keys of the parent rows.

where `GetRows()` method in ADO retrieves multiple records from a Recordset object and stores them in a two-dimensional array.

The `PutItems` method loads items from a safe array. The `PutItems` method may raise one of the following exceptions:

- **The array dimension exceeds 2** (In simpler terms, a two-dimensional array (or 2D array) is like a table with rows and columns. If an array exceeds 2 dimensions, it means it has three or more dimensions, such as a 3D array (which can be thought of as a collection of tables) or even higher dimensions) You need to provide a one-dimensional or two-dimensional array
- **The number of columns does not match the array size** (either the control has no columns or the number of columns is too small). You need to add more columns ([Add](#) property).
- **The element type of the array is not valid** (the type of the array is either unknown or not supported) You need to provide a valid type, which must be one of the following: Variant, String, Integer, Long, Double, Float, or Date.

The PutItems method performs:

1. **Insertion Order:** The data is inserted into the system in the same order as it is provided by the Items parameter. This means that the sequence of rows in the Items parameter directly affects how the data is inserted.
2. **Sorting Requirement:** To ensure correct insertion, it's crucial that the data is sorted by the IDColumn (when the Parent parameter is of "parent;IDColumn;ParentIDColumn" format). This sorting ensures that parent rows are inserted before their corresponding child rows.
3. **Parent-Child Relationship:** The sorting ensures that when a row refers to a parent row using the ParentIDColumn value (when the Parent parameter is of "parent;IDColumn;ParentIDColumn" format). The parent row is already present in the control. This allows the current row to be inserted as a child of the parent row without encountering errors or inconsistencies.

In essence, by sorting the data appropriately, you establish a clear hierarchy where parent rows are inserted before child rows, maintaining the integrity of the parent-child relationships within the dataset.

For instance, let's say we have the following data:

EmployeeID	EmployeeName	DepartmentID	ParentID
1	John	101	
2	Alice	102	1
3	Bob	101	1
4	Sarah	102	1
5	Emma	101	2
6	Mike	102	2

Each row represents an employee.

- EmployeeID uniquely identifies each employee (represents the column with the index 0)
- EmployeeName denotes the name of the employee (represents the column with the index 1)
- DepartmentID indicates the department to which the employee belongs (represents the column with the index 2)
- ParentID establishes the relationship between employees (represents the column with the index 3), where it references the EmployeeID of the parent employee. An empty value indicates the absence of a parent, typically representing the head of the department.

Having this data organized into a two-dimensional array, the statement `PutItems d` loads it as a flat table:

EmployeeID	EmployeeName	DepartmentID	ParentID
1	John	101	
2	Alice	102	1
3	Bob	101	1
4	Sarah	102	1
5	Emma	101	2
6	Mike	102	2

whereas `PutItems d, ";0;3"` loads it as a group structure:

EmployeeID	EmployeeName	DepartmentID	ParentID
1	John	101	
2	Alice	102	1
5	Emma	101	2
6	Mike	102	2
3	Bob	101	1
4	Sarah	102	1

where `d` is an array as defined next:

```
Dim d(3, 5) As Variant
```

```
d(0, 0) = "1": d(1, 0) = "John": d(2, 0) = "101": d(3, 0) = ""
```

```
d(0, 1) = "2": d(1, 1) = "Alice": d(2, 1) = "102": d(3, 1) = "1"
```

```
d(0, 2) = "3": d(1, 2) = "Bob": d(2, 2) = "101": d(3, 2) = "1"
```

```
d(0, 3) = "4": d(1, 3) = "Sarah": d(2, 3) = "102": d(3, 3) = "1"
```

```
d(0, 4) = "5": d(1, 4) = "Emma": d(2, 4) = "101": d(3, 4) = "2"
```

```
d(0, 5) = "6": d(1, 5) = "Mike": d(2, 5) = "102": d(3, 5) = "2"
```

Use the [GetItems](#) method to get a safe array with the items in the control. The `PutItems` method fires [AddItem](#) event for each item added to `Items` collection. Use the [Items](#) property to access the items collection. 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 Group.RadiolImage(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.

Use RadiolImage and [CheckImage](#) properties to define the icons used for radio and check box cells.

The following sample shows how to change the default icon for cells of radio type:

```
Group.RadiolImage(True) = 1      ' Sets the icon for cells of radio type that are checked
Group.RadiolImage(False) = 2    ' Sets the icon for cells of radio type that are
unchecked
```

The Group.RadiolImage(True) = 0 makes the group to use the default icon for painting cells of radio type that are checked.

property Group.RClickSelect as Boolean

Retrieves or sets a value that indicates whether an item is selected using right mouse button.

Type	Description
Boolean	A boolean expression that indicates whether an item is selected using the right mouse button.

Use the RClickSelect property to allow selecting items by right click. By default, the RClickSelect property is False.

method **Group.Refresh ()**

Refreshes the group's content.

Type	Description
------	-------------

The Refresh method forces repainting the group.

property Group.RightToLeft as Boolean

Indicates whether the group 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 RightToLeft 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")

property Group.ScrollBars as ScrollBarsEnum

Returns or sets a value that determines whether the group has horizontal and/or vertical scroll bars.

Type	Description
ScrollBarsEnum	A ScrollBarsEnum expression that identifies which scroll bars are visible.

Use the ScrollBars property to disable the group's scroll bars. By default, the ScrollBars property is exBoth, so both scroll bars are used if necessarily. For instance, if the ScrollBars property is exNone the group displays no scroll bars. If the [AutoHeight](#) property is True, the group displays no vertical scroll bar. The exVertical flag is removed if the AutoHeight property is changing at runtime. You can call the ScrollBars property **after** setting the AutoHeight property, in case you need vertical scroll bar. The control displays a scroll bar only if it is required. Use the [AutoScrollBar](#) property to hide the control's vertical scroll bar.

property Group.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 [ScrollBars](#) property to specify which scroll bar is visible or hidden in the control. 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 Group.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 [ScrollBars](#) property to specify which scroll bar is visible or hidden in the control. 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 Group.ScrollBySingleLine as Boolean

Retrieves or sets a value that indicates whether the group scrolls the lines to the end, item by item.

Type	Description
Boolean	A boolean expression that indicates whether the group scrolls the lines to the end, item by item.

We recommend to set the ScrollBySingleLine property if you have one of the following:

- If you have at least a cell that has [CellSingleLine](#) property on false
- If the group contains items with different heights.
- If your group contains at least an item that hosts an ActiveX group. See [InsertControlItem](#) property.

property Group.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

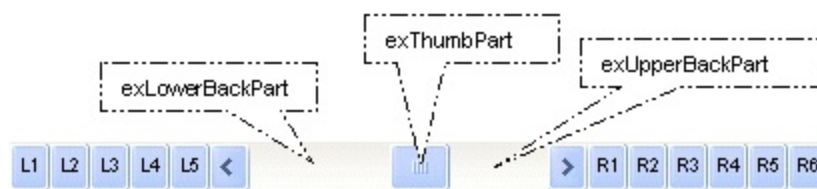
Use the ScrollFont property to specify the font in the control's scroll bar. Use the [ScrollPartCaption](#) property to specify the caption of the scroll's part. Use the [ScrollPartVisible](#) property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, 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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar.

property Group.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 displayed
String	A String expression that specifies the caption being displayed on the part of the scroll bar.

Use the ScrollPartCaption 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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar. Use the [ScrollFont](#) property to specify the font in the control's 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)

property Group.ScrollPartCaptionAlignment(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as AlignmentEnum

Specifies the alignment of the caption in the part of the scroll bar.

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 displayed
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the caption in the part of the scrollbar.

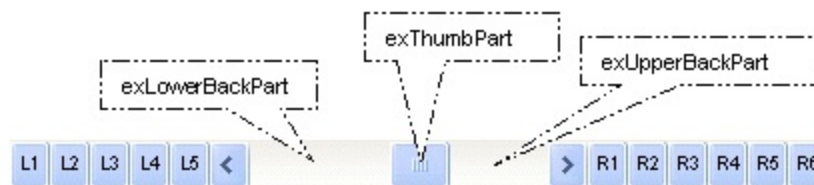
The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar. By default, the caption is centered. Use the [ScrolPartCaption](#) property to specify the caption being displayed on specified part of the scroll bar. Use the [ScrollPartVisible](#) property to add or remove buttons/parts in the control's scrollbar.

property Group.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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [ScrolPartCaption](#) property to specify the caption of the scroll's part. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar.

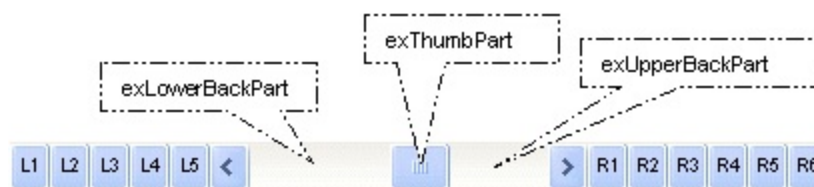


property Group.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

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 [ScrollBars](#) property to specify the visible scrollbars in the control. Use the [ScrolPartCaption](#) property to specify the caption of the scroll's part. Use the [OffsetChanged](#) event to notify your application that the scroll position is changed. Use the [OversizeChanged](#) event to notify your application whether the range for a specified scroll bar is changed. The control fires the [ScrollButtonClick](#) event when the user clicks a part of the scroll bar. Use the [Background](#) property to change the visual appearance for any part in the control's 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)

property Group.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\(exVSTThumb\)](#) or [Background\(exHSTThumb\)](#) 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 Group.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. The [OffsetChanged](#) event notifies your application that the user changes the scroll position. Use the [SortPartVisible](#) property to specify the parts being visible in the control's scroll bar. Use the [ScrollBars](#) property to specify the visible scrollbars in the control.

property Group.SearchColumnIndex as Long

Retrieves or sets a value indicating the column's index that is used for auto search feature.

Type	Description
Long	A long expression indicating the column's index that is used for auto search feature.

The SearchColumnIndex is changed if the user press TAB or Shift + TAB.

property Group.SelBackColor as Color

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

Type	Description
Color	A color expression that indicates the selection background color.

Use the SelBackColor and [SelForeColor](#) properties to define the colors used for selected items.

property Group.SelBackMode as BackModeEnum

Retrieves or sets a value that indicates whether the selection is transparent or opaque.

Type	Description
BackModeEnum	A BackModeEnum expression that indicates whether the selection is transparent or opaque.

Use the SelBackMode property to specify how the selection appears. Use the SelBackMode property to specify how the group displays the selection when the group has a [picture](#) on its background.

property Group.SelectColumn as Boolean

Specifies whether the user selects cells only in SelectColumnIndex column, while FullRowSelect property is False.

Type	Description
Boolean	A boolean expression that specifies whether the user selects cells only in SelectColumnIndex column, while the FullRowSelect property is False

By default, the SelectColumn property is False. The SelectColumn property has effect only if the FullRowSelect is False. The group displays the selected cell in the SelectColumnIndex column. The SelectColumnIndex property specifies the index of selected column.

property Group.SelectColumnIndex as Long

Retrieves or sets a value that indicates the column's index where the user can select an item by clicking.

Type	Description
Long	A long expression that indicates the column's index where the user can select the item.

The property has effect only if the [FullRowSelect](#) property is False.

property Group.SelectColumnInner as Long

Retrieves or sets a value that indicates the index of the inner cell that's selected.

Type	Description
Long	A long expression that indicates the index of the inner cell that's focused or selected.

Use the `SelectColumnInner` property to get the index of the inner cell that's selected or focused. The `SelectColumnInner` property may be greater than zero, if the control contains inner cells. The [SplitCell](#) method splits a cell in two cells. The newly created cell is called inner cell. The [FocusItem](#) property indicates the focused item. The [SelectColumnIndex](#) property determines the index of the column that's selected when [FullRowSelect](#) property is `False`.

property Group.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.

Use the SelForeColor and [SelBackColor](#) properties to change the colors used for selected items.

property Group.SelLength as Long

Returns or sets the number of characters selected.

Type	Description
Long	A long expression that indicates the number of characters selected.

The [SelStart](#) and SelLength properties are valid only if the group's textbox field is visible. Use the [AllowEdit](#) property to allow group edits the data using a text box field. Use the [Edit](#) method to start editing a cell using a textbox field.

property Group.SelStart as Long

Returns or sets the starting point of text selected; indicates the position of the insertion point if no text is selected.

Type	Description
Long	A long expression that indicates the starting point of text selected

The SelStart and [SelLength](#) properties are valid only if the group's textbox field is visible. Use the [AllowEdit](#) property to allow group edits the data using a text box field. Use the [Edit](#) method to start editing a cell using a textbox field.

method Group.SetFocus ()

Sets the keyboard focus to the group's list window.

Type	Description
------	-------------

Use the [hWnd](#) property to access the group's list window. Use the SetFocus to focus the group's list window.

The following sample sets the keyboard focus to the group that's expanded or collapsed:

```
Private Sub ExplorerTree1_SelectGroup(ByVal Group As EXPLORERTREELibCtl.IGroup)
    Group.SetFocus
End Sub
```


property Group.Shortcut as String

Specifies the name of the shortcut which displays the group.

Type	Description
String	A HTML expression that indicates the caption of the shortcut.

The Group objects with the same Shortcut property belongs to the same set, and displays the Shortcut caption in the control's shortcut bar. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. By default, the Shortcut property is empty, so all Group in the Groups collection belongs to the same set. The shortcut bar displays the first icon in the HTML caption, if found, or it displays a custom size picture if specified using the the [ShortcutPicture](#) property. If the Shortcut has associated a custom size picture ([ShortcutPicture](#) property), the first icon found in the HTML caption is not displayed in the shortcut bar. The entire Shortcut caption is displayed when the shortcut is expanded. Use the [ExpandShortcutCount](#) property to expand the number of shortcuts in the control's shortcut bar. The [ExpandShortcut](#) event notifies your application when the user resizes the control's shortcut bar. The control fires the [SelectShortcut](#) event when the user selects a shortcut, so groups that belongs to the shortcut are displayed.

The Shortcut property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- **<i> ... </i>** 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.

The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.




- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu

" that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** 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=rrggb> ... </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=rrggb> ... </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=rrggb> ... </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=rrggb> ... </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
- **number[:width]** 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.

- **key[:width]** 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 FFFFFFFF;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 Group.ShowFocusRect as Boolean

Retrieves or sets a value indicating whether the group draws a thin rectangle around the focused item.

Type	Description
Boolean	A boolean expression that indicates whether the group draws a thin rectangle around the focused item.

Use the ShowFocusRect property to hide the rectangle drawn around the focused item.

property Group.ShowLockedItems as Boolean

Retrieves or sets a value that indicates whether the control displays the locked items.

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 [CellCaption](#) property to specify the caption for a cell.

property Group.SingleSel as Boolean

Retrieves or sets a value that indicates whether the group supports single or multiple selection.

Type	Description
Boolean	A boolean expression that indicates whether the group supports single or multiple selection.

Use the SingleSel property to enable multiple selection.

property Group.SortOnClick as SortOnClickEnum

Retrieves or sets a value that indicates whether the group sorts automatically the data when the user click on column's caption.

Type	Description
SortOnClickEnum	A SortOnClick expression that indicates whether the group sorts automatically the data when the user click on the column's caption.

Use the SortOnClick property to disable sorting items when the user clicks on the column's header.

There are two methods to get the items sorted like follows:

- Using the SortOrder property of the [Column](#) object::

```
Group.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 group. 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 group displays a list, all items are considered being root items).

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


property Group.StrikeOut as Boolean

Specifies a value that indicates whether the group's caption should appear in strikeout.

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

Use the [Bold](#), [Italic](#), [Underline](#) and [StrikeOut](#) properties to customize the caption's font attributes.

property Group.ToolTip as Variant

Specifies the group's tooltip.

Type	Description
Variant	A string expression that indicates the group's tooltip. The ToolTip supports built-in HTML format as described here .

The ToolTip shows up when the cursor hovers the group's caption. The [ToolTipPopDelay](#) property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the [ToolTipWidth](#) property to specify the width of the tooltip window. Use the [Background\(exToolTipAppearance\)](#) property indicates the visual appearance of the borders of the tooltips. Use the [Background\(exToolTipBackColor\)](#) property indicates the tooltip's background color. Use the [Background\(exToolTipForeColor\)](#) property indicates the tooltip's foreground color. Use the [CellToolTip](#) property to specify the cell's tooltip. Use the [ShowToolTip](#) method to programmatically display a custom tooltip.

property Group.Top as Long

Specifies the distance between the top edge of the control and group's list.

Type	Description
Long	A long expression that indicates the distance between the top edge of the control and group's list.

The Left, Top, Width and Height properties determines the location of the group's list in client coordinates. Use the [LayoutChanged](#) event and [Left](#), [Top](#), [Width](#) and [Height](#) to determine the coordinates of the group's list on the fly.

property Group.TreeColumnIndex as Long

Retrieves or sets a value indicating the column's index where the hierarchy will be displayed.

Type	Description
Long	A long expression that indicates the index of the column where the group's hierarchy is displayed.

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.

property Group.Underline as Boolean

Specifies a value that indicates whether the group's caption is underlined.

Type	Description
Boolean	A boolean expression that specifies whether the group's caption is underlined.

Use the [Bold](#), [Italic](#), [Underline](#) and [StrikeOut](#) properties to customize the caption's font attributes.

property Group.UserData as Variant

Specifies an extra data.

Type	Description
Variant	A Variant that indicates the item's extra data

The UserData property associates an extra data to the item. The UserData property is not used by the group or control.

property Group.Width as Long

Retrieves the width in pixels of the group's list.

Type	Description
Long	A long expression that indicates the width in pixels of the group's list

The Left, Top, Width and Height properties determines the location of the group's list in client coordinates. Use the [IndentGroupLeft](#) and [IndentGroupRight](#) properties to indent the group's list to the left or right side. Use the [LayoutChanged](#) event and [Left](#), [Top](#), Width and [Height](#) to determine the coordinates of the group's list on the fly.

Groups object

The Groups collection holds [Group](#) objects. Each Group holds a collection of columns and items that displays data as a tree or list as well. Use the [Items](#) property to access the group's [Items](#) collection. Use the [Columns](#) property to access the group's [Columns](#) collection. Use the [AddItem](#) method to add new items to the group. Use the [Add](#) method to add new columns to the group.

Name	Description
Add	Adds a Group object to the collection and returns a reference to the newly created object.
Clear	Removes all objects in the collection.
Count	Returns the number of objects in the collection.
Item	Returns a specific Group from the collection.
ItemByPos	Retrieves the group given its position.
Remove	Removes a specific member from the collection.

method Groups.Add (Caption as String)

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

Type	Description
Caption as String	A string expression that indicates the group's caption. The Caption supports built-in HTML format as described here , if the CaptionFormat property is exHTML.
Return	Description
Group	A Group object being added to Groups collection.

The Add method adds a new Group object to the Groups collection. The [AddGroup](#) event is fired each time when a new group is added to Groups collection. Use the [Items](#) property to access the group's [Items](#) collection. Use the [Columns](#) property to access the group's [Columns](#) collection. Use the [AddItem](#) method to add new items to the group. Use the [Add](#) method to add new columns to the group. The caption may contain built-in HTML tags, if the [CaptionFormat](#) property is exHTML. Use the [Caption](#) property to access the group's caption.

method **Groups.Clear ()**

Removes all objects in the collection.

Type	Description
------	-------------

Clears the Groups collection. Use the [Remove](#) method to remove a specific group. The [RemoveGroup](#) event is fired when user removes a group.

property Groups.Count as Long

Returns the number of objects in the collection.

Type	Description
Long	A long expression that specifies the count of Group objects into Groups collection.

Counts the Group objects in the Groups collection.

property Groups.Item (Index as Variant) as Group

Returns a specific Group from the collection.

Type	Description
Index as Variant	A long expression that indicates the group's index, or a string expression that indicates the group's caption.
Group	A Group object being retrieved.

use the Item property to access a given Group object. The Item property is the default property in the Groups object, and Groups.Item(x) is similar with Groups(x). Use the [ItemByPos](#) property to access a Group object by its position.

property Groups.ItemByPos (Position as Long) as Group

Retrieves the group given its position.

Type	Description
Position as Long	A long expression that indicates the position of the requested group
Group	A Group object being accessed by its position.

Use the ItemByPos property to access the Group object by its position.

method `Groups.Remove (Index as Variant)`

Removes a specific member from the collection.

Type	Description
Index as Variant	A long expression that indicates the the group's index, or a string expression that indicates the group's caption.

Use the `Remove` method to remove a specific `Group` object. The [RemoveGroup](#) event is fired when the user removes a group. Use the [Clear](#) method to clear the entire `Groups` 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 group. To access the Items collection use [Items](#) property of the control. Using the Items collection you can add, remove or change the items in the group. 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	Retrieves a value indicating whether the SetParent method can be accomplished..
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 indicates whether the cell's caption should appear in bold.
CellButtonAutoWidth	Retrieves or sets a value indicating whether the cell's button fits the cell's caption.
CellCaption	Retrieves or sets the text displayed on a specific cell.
CellCaptionFormat	Specifies how the cell's caption is displayed.
CellChecked	Retrieves the cell's handle that is checked on a specific radio group.
CellData	Retrieves or sets the extra data for a specific cell.
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.
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 or not.
CellHasCheckBox	Retrieves or sets a value indicating whether the cell has associated a checkbox or not.
CellHasRadioButton	Retrieves or sets a value indicating whether the cell has associated a radio button or not.
CellHyperLink	Specifies whether the cell's is highlighted when the cursor

mouse is over the cell.

[CellImage](#)

Retrieves or sets an Image that is displayed on the cell's area.

[CellImages](#)

Specifies an additional list of icons shown in the cell.

[CellItalic](#)

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

[CellItem](#)

Retrieves the handle of item that is the owner of a specific 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 a value that indicates the Picture object displayed by the cell.

[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.

[CellState](#)

Retrieves or sets the cell's state. Has effect only for check and radio cells.

[CellStrikeOut](#)

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

[CellToolTip](#)

Retrieves or sets a text that is used to show the tooltip's cell.

[CellUnderline](#)

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

[CellVAlignment](#)

Retrieves or sets a value that indicates how the cell's caption is vertically aligned.

[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.
DefaultItem	Retrieves or sets the default item.
Edit	Edits a cell.
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 Caption in ColIndex colum. The searching starts at StartIndex item.
FindItemData	Finds the item giving its data.
FindPath	Finds the item, given its path. The group searches the path on the SearchColumnIndex column.
FirstVisibleItem	Retrieves the handle of the first visible item in the group.
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 group. The caption is taken from the column SearchColumnIndex.
InnerCell	Retrieves the inner cell.
InsertControllItem	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.
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 based on a specific column.
ItemChild	Retrieves the child of a specified item.
ItemControllID	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.
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.
ItemObject	Retrieves the ActiveX object associated, if the item was created using InsertControlItem method.
ItemParent	Returns the handle of 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 into 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 InsertControllItem.
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/fixed item.
LockedItemCount	Specifies the number of items fixed on the top or bottom side of the control.
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 group.
RemoveItem	Removes a specific item.
RootCount	Retrieves the number of root objects into Items collection.
RootItem	Retrieves the handle of the root item giving its index into 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.
SelectItem	Selects or unselects a specific item.
SetParent	Changes the parent of the given item.
SortableItem	Specifies whether the item is sortable.

[SortChildren](#)

Sorts the childitems of the given parent item in the group. SortChildren will not recurse through the tree, 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

Retrieves a value indicating whether the SetParent method can be accomplished.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being moved.
NewParent as HITEM	A long expression that indicates the handle of the parent item where the item should be moved.
Boolean	A boolean expression that indicates whether the item can be child of the NewParent item.

Use this property to make sure that [SetParent](#) can be called.

method Items.AddItem ([Caption as Variant])

Adds a new item, and returns a handle to the newly created item.

Type	Description
Caption as Variant	A string expression that indicates the cell's caption for the first column. or a safe array that contains the captions for each column. The Caption accepts HTML format, if the CellCaptionFormat property is exHTML.
Return	Description
HITEM	A long expression that indicates the handle of the newly created item.

Use the AddItem property when your group acts like a list. Use [InsertItem](#) when your group looks like a tree. Use the [InsertControlItem](#) property when the item needs to host an ActiveX control. The AddItem property adds a new item that has no parent. When a new item is added (inserted) to the [Items](#) collection, the control fires the [AddItem](#) event. If the group contains more than one column use the [CellCaption](#) property to set the cell's caption. If there are no columns the AddItem method fails. By default, the group adds a default column. Use the [LockedItemCount](#) property to lock or unlock items to the top or bottom side of the group. Use the [MergeCells](#) method to combine two or more cells in a single cell. Use the [SplitCell](#) property to split a cell.

The following sample uses the VB Array function to add two items:

```
With ExplorerTree1
  With .Groups.Add("Group 1")
    .BeginUpdate

    .Columns.Add "Column 1"
    .Columns.Add "Column 2"
    .Columns.Add "Column 3"

    With .Items
      .AddItem Array("Item 1.1", "Item 1.2", "Item 1.3")
      .AddItem Array("Item 2.1", "Item 2.2", "Item 2.3")
    End With

    .EndUpdate
  End With
```

Use the [PutItems](#) method to load an array, like in the following sample:

```
With ExplorerTree1
```

```
  With .Groups.Add("Group 1")
```

```
    Set rs = CreateObject("ADODB.Recordset")
```

```
    rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
```

```
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
```

```
    .ColumnAutoResize = False
```

```
    .HeaderVisible = True
```

```
    .BeginUpdate
```

```
      ' Add the columns
```

```
      With .Columns
```

```
        ' By default the control adds a column, so we delete it first
```

```
        .Clear
```

```
        For Each f In rs.Fields
```

```
          .Add f.Name
```

```
        Next
```

```
      End With
```

```
      .PutItems rs.getRows()
```

```
    .EndUpdate
```

```
  End With
```

```
End With
```

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 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. 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.

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.

For instance, the following code shows how to change the left top cell of your group:
`Group.Items.CellBackColor(Group.Items(0), 0) = vbBlue`

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see [Column.Index](#) property) of a column , the column's caption (a string value, see [Column.Caption](#) property), or a handle to a cell (see [ItemCell](#) property). Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```


property Items.CellBold([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that indicates whether the cell's caption 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 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.

Here's a snippet of code that shows how to bold the first column in the first group(it enumerates all cells in the column):

```
Dim h As Variant
With ExplorerTree1.Groups(0)
    .BeginUpdate
    With .Items
        For Each h In ExplorerTree1.Groups(0).Items
            .CellBold(h, 0) = True
        Next
    End With
    .EndUpdate
End With
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

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 column's index, or 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 whether all buttons in the column fit the cell's content.

property Items.CellCaption([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves or sets the text displayed on a specific 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, or the handle to the cell, if the Item parameter is 0, a string expression that indicates the column's caption or the column's key.
Variant	A variant expression that indicates the cell's caption. The cell's caption supports built-in HTML format.

The CellCaption property specifies the cell's caption. To associate an user data for a cell you can use [CellData](#) property. Use the [CellCaptionFormat](#) property to use HTML tags in the cell's caption. Use the [ItemData](#) property to associate an extra data to an item. To hide a column you have to use [Visible](#) property of the [Column](#) object. The [AddItem](#) method specifies also the caption for the first cell in the item. Use the [SplitCell](#) property to split a cell.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see [Column.Index](#) property) of a column , the column's caption (a string value, see [Column.Caption](#) property), or a handle to a cell (see [ItemCell](#) property). Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True  
Group.Items.CellBold(Group.Items(0), 0) = True  
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellCaptionFormat([Item as Variant], [ColIndex as Variant]) as CaptionFormatEnum

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 cell's handle, or a string expression that specifies the column's caption
CaptionFormatEnum	A CaptionFormatEnum expression that defines the way how the cell's caption is displayed.

The component supports built-in HTML format. That means that you can use HTML tags when displays the cell's caption . By default, the CellCaptionFormat property is exText. If the CellCaptionFormat is exText, the cell displays the [CellCaption](#) property like it is. If the CellCaptionFormat is exHTML, the cell displays the CellCaption property using the HTML tags specified in the CaptionFormatEnum type.

property Items.CellChecked (RadioGroup as Long) as HCELL

Retrieves the cell's handle that is checked on a specific radio group.

Type	Description
RadioGroup as Long	A long expression that indicates the radio group identifier.
HCELL	A long expression that identifies the handle of the cell that's checked in the specified radio group. To retrieve the handle of the owner item you have to use CellItem property.

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. The following sample group all cells of the first column into a radio group, and display the cell's checked on the radio group when the state of a radio group has been changed:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal
Item As EXPLOREERTREELibCtl.HITEM)
    With Group.Items
        .CellHasRadioButton(Item, 0) = True
        .CellRadioGroup(Item, 0) = 1234 ' The 1234 is arbitrary and it represents the identifier
for the radio group
    End With
End Sub

Private Sub ExplorerTree1_CellStateChanged(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)
    Debug.Print "In the 1234 radio group the """" & Group.Items.CellCaption(
Group.Items.CellChecked(1234)) & """" is checked."
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
```

```
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellData([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves or sets the extra data for a specific 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 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/group, it is only for user use.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```


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 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.

Once that one cell is disabled it cannot be checked or clicked. To disable a column you can use [Enabled](#) property of the Column object.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see [Column.Index](#) property) of a column , the column's caption (a string value, see [Column.Caption](#) property), or a handle to a cell (see [ItemCell](#) property). Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True  
Group.Items.CellBold(Group.Items(0), 0) = True  
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

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, or optional if the cell's handle is passed to ColIndex parameter
ColIndex as Variant	A long expression that indicates the column's index or cell's handle, or a string expression that indicates the column's caption.
IFontDisp	A Font object that indicates the cell's font.

By default, the CellFont property is nothing. If the CellFont property is nothing, 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 [CellCaptionFormat](#) to specify different font attributes.

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 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. To change the background color for an item you can use [ItemBackColor](#) property.

For instance, the following code shows how to change the left top cell of your control:
`Group.Items.CellForeColor(Group.Items(0), 0) = vbBlue`

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True  
Group.Items.CellBold(Group.Items(0), 0) = True  
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

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.

property Items.CellHasButton([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has associated a push button or not.

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.
Boolean	A boolean expression that indicates whether the cell contains a button.

When cell's button is clicked the control fires [CellButtonClick](#) event. The caption of the push button is specified by the [CellCaption](#) property.

The following sample sets the cells of the first column to be of button type, and displays a message if the button is clicked:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal Item As EXPLOREERTREELibCtl.HITEM)
    With Group.Items
        .CellHasButton(Item, 0) = True
    End With
End Sub

Private Sub ExplorerTree1_CellButtonClick(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)
    MsgBox "The cell '" & Group.Items.CellCaption(Item, ColIndex) & "' has been clicked"
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellHasCheckBox([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has associated a checkbox or not.

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.
Boolean	A boolean expression that indicates whether the cell contains a check box button.

To change the state for a check cell you have to use [CellState](#) property. The cell cannot display in the same time a radio and a check button. The control fires [CellStateChanged](#) event when the cell's state has been changed. To set the cell of radio type you have call [CellHasRadioButton](#) property. Use the [FilterType](#) property on exCheck to filter for checked or unchecked items.

The following sample shows how to set the check type for all cells of the first column in the first group:

```
Dim h As Variant
With ExplorerTree1.Groups(0)
    .BeginUpdate
    With .Items
        For Each h In ExplorerTree1.Groups(0).Items
            .CellHasCheckBox(h, 0) = True
        Next
    End With
    .EndUpdate
End With
```

Another sample that shows how to set the type of cells to radio is:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal
Item As EXPLORERTREELibCtl.HITEM)
    With Group.Items
        .CellHasCheckBox(Item, 0) = True
    End With
```

End Sub

The following sample shows how to use the `CellStateChanged` event to display a message when a cell of radio or check type has changed its state:

```
Private Sub ExplorerTree1_CellStateChanged(ByVal Group As EXPLORERTREELibCtl.IGroup,
ByVal Item As EXPLORERTREELibCtl.HITEM, ByVal ColIndex As Long)
    Debug.Print "The cell """" & Group.Items.CellCaption(Item, ColIndex) & """" has changed
its state. The new state is " & If(Group.Items.CellState(Item, ColIndex) = 0, "Unchecked",
"Checked")
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an `Item` and a `ColIndex` parameters are referring to a cell. The `Item` parameter represents the handle of an item, and the `ColIndex` parameter indicates an index (a numerical value, see `Column.Index` property) of a column , the column's caption (a string value, see `Column.Caption` property), or a handle to a cell. Here's few hints how to use properties with `Item` and `ColIndex` parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```


property Items.CellHasRadioButton([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has associated a radio button or not.

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.
Boolean	A boolean expression that indicates whether the cell contains a radio button.

Retrieves or sets a value indicating whether the cell has associated a radio button or not. To change the state for a radio cell you have to use [CellState](#) property. The cell cannot display in the same time a radio and a check button. The control fires [CellStateChanged](#) event when the cell's state has been changed. To set the cell of check type you have call [CellHasCheckBox](#) property. To add or remove a cell to a given radio group you have to use [CellRadioGroup](#) property. The following sample shows how to set the radio type for all cells of the first column, and group all of them in the same radio group (1234):

```
Dim h As Variant
With ExplorerTree1.Groups(0)
    .BeginUpdate
    With .Items
        For Each h In ExplorerTree1.Groups(0).Items
            .CellHasRadioButton(h, 0) = True
            .CellRadioGroup(h, 0) = 1234
        Next
    End With
    .EndUpdate
End With
```

or

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal
Item As EXPLORERTREELibCtl.HITEM)
    With Group.Items
        .CellHasRadioButton(Item, 0) = True
    End With
End Sub
```

```
.CellRadioGroup(Item, 0) = 1234
```

```
End With
```

```
End Sub
```

To find out the radio cell that is checked in the radio group 1234 you have to call: [MsgBox](#)
Group.Items.CellCaption(, Group.Items.CellChecked(1234))

The following sample group all cells of the first column into a radio group, and display the cell's checked on the radio group when the state of a radio group has been changed:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal  
Item As EXPLOREERTREELibCtl.HITEM)
```

```
With Group.Items
```

```
.CellHasRadioButton(Item, 0) = True
```

```
.CellRadioGroup(Item, 0) = 1234
```

```
End With
```

```
End Sub
```

```
Private Sub ExplorerTree1_CellStateChanged(ByVal Group As EXPLOREERTREELibCtl.IGroup,  
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)
```

```
Debug.Print "In the 1234 radio group the "" & Group.Items.CellCaption(  
Group.Items.CellChecked(1234)) & "" is checked."
```

```
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
```

```
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellHyperLink ([Item as Variant], [ColIndex as Variant]) as Boolean

Specifies whether the cell's 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 column's index, or a string expression that indicates the column's caption.
Boolean	A boolean expression that indicates whether the cell is highlighted when the cursor is over the cell.

Use the CellHyperLink property to add hyperlink cells to your group. Use the [HyperLinkClick](#) event to notify your application when a hyperlink cell is clicked. Use the [CellForeColor](#) property to specify the cell's foreground color. Use the [HyperLinkColor](#) property to specify the hyperlink color.

property Items.CellImage ([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets an Image that is displayed on the cell's area.

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 value that indicates the image index.

Use the CellImage property to assign a single icon to a cell. Use the [CellImages](#) property to assign multiple icons to a cell. Use the [Images](#) method to assign icons to the control at runtime. You can add images at design time by dragging a file to image editor of the control. The CellImage = 0 removes the cell's image. The collection of Images is 1 based. The [CellImageClick](#) event occurs when the cell's image is clicked. Use the [FilterType](#) property on exImage to filter items by icons. Use the [CellPicture](#) property to load a custom size picture to a cell. Use the **** HTML tag to insert icons inside the cell's caption, if the [CellCaptionFormat](#) property is exHTML.

The following sample sets cell's image for the first column (to run the sample make sure that control's images collection is not empty):

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal Item As EXPLOREERTREELibCtl.HITEM)
    With Group.Items
        .CellImage(Item, 0) = 1
    End With
End Sub
```

The following sample changes the cell's image when the user has clicked on the cell's image (to run the following sample you have to add two images to the tree's images collection.),

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal Item As EXPLOREERTREELibCtl.HITEM)
    With Group.Items
        .CellImage(Item, 0) = 1
    End With
```

```
End Sub
```

```
Private Sub ExplorerTree1_CellImageClick(ByVal Group As EXPLOREERTREELibCtl.IGroup,  
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)
```

```
    With Group.Items
```

```
        .CellImage(Item, ColIndex) = .CellImage(Item, ColIndex) Mod 2 + 1
```

```
    End With
```

```
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
```

```
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

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.

The CellImages property assigns multiple icons to a 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.

The following sample assigns the first and the third icon to the cell:

```
With ExplorerTree1.Groups(0).Items
    .CellImages(.ItemByIndex(0), 1) = "1,3"
End With
```

property Items.CellItalic([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that indicates whether the cell's caption 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 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.

The CellItalic property specifies whether the cell should appear in italic. To change the italic attribute for the whole item call [ItemItalic](#) property.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellItem (Cell as HCELL) as HITEM

Retrieves the handle of the item that is owner for a specific cell.

Type	Description
Cell as HCELL	A long expression that indicates the handle of the cell.
HITEM	A long expression that indicates the handle of the item.

Use the CellItem property to retrieve the item's handle. Use the [ItemCell](#) property to get the cell's handle given an item and a column.

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.

The following sample shows few methods to unmerge cells:

```
With ExplorerTree1.Groups(0)
  With .Items
    .UnmergeCells .ItemCell(.RootItem(0), 0)
  End With
End With
```

```
With ExplorerTree1.Groups(0)
  With .Items
    Dim r As Long
    r = .RootItem(0)
    .UnmergeCells Array(.ItemCell(r, 0), .ItemCell(r, 1))
  End With
End With
```

```
With ExplorerTree1.Groups(0)
  .BeginUpdate
  With .Items
```

```
.CellMerge(.RootItem(0), 0) = -1
```

```
.CellMerge(.RootItem(0), 1) = -1
```

```
.CellMerge(.RootItem(0), 2) = -1
```

```
End With
```

```
.EndUpdate
```

```
End With
```

You can merge the first three cells in the root item using any of the following methods:

```
With ExplorerTree1.Groups(0)
```

```
With .Items
```

```
.CellMerge(.RootItem(0), 0) = Array(1, 2)
```

```
End With
```

```
End With
```

```
With ExplorerTree1.Groups(0)
```

```
.BeginUpdate
```

```
With .Items
```

```
Dim r As Long
```

```
r = .RootItem(0)
```

```
.CellMerge(r, 0) = 1
```

```
.CellMerge(r, 0) = 2
```

```
End With
```

```
.EndUpdate
```

```
End With
```

```
With ExplorerTree1.Groups(0)
```

```
.BeginUpdate
```

```
With .Items
```

```
Dim r As Long
```

```
r = .RootItem(0)
```

```
.MergeCells .ItemCell(r, 0), .ItemCell(r, 1)
```

```
.MergeCells .ItemCell(r, 0), .ItemCell(r, 2)
```

```
End With
```

```
.EndUpdate
```

```
End With
```

```
With ExplorerTree1.Groups(0)
```

With .Items

Dim r As Long

r = .RootItem(0)

.MergeCells .ItemCell(r, 0), **Array**(.ItemCell(r, 1), .ItemCell(r, 2))

End With

End With

With ExplorerTree1.Groups(0)

With .Items

Dim r As Long

r = .RootItem(0)

.MergeCells **Array**(.ItemCell(r, 0), .ItemCell(r, 1), .ItemCell(r, 2))

End With

End With

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. The CellParent property gets 0 if the cell is 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 a value that indicates the Picture object displayed by 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 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 group 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](#), [CellImages](#) property to associate an icon from the control's [Images](#) collection.

The following sample shows how to load a picture to a cell:

```
Group.Items.CellPicture(h, 0) = LoadPicture("c:\winnt\logo.gif")
```

The following sample associates a picture to a cell by loading a base64 encoded string:

```
Dim s As String
s =
"gBCJr+BAAg0HGwEgwog4jg4ig4BAEFg4AZEKisZjUbAAzg5mg6Zg7Mg7/g0ek8oGcgjsijsk

s = s +
"XgBadIDXdYSXRb9wWBclK2taF1gAI5HiPaN8oPdINWbaF23KAwyWkNYyXxg9p3WNYjU/c


With ExplorerTree1.Groups(1).Items
    .CellPicture(.ItemByIndex(0), 0) = s
    .ItemHeight(.ItemByIndex(0)) = 24
End With
```

Mail

 **Inbox** (23)

Unread Email (11)

To Follow Up (7)

 My Folders

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. 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. 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 the radio group where the cell is contained.

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 value that identifies the cell's radio group.

A radio cell cannot be contained by two different radio groups. When a cell state is changed the control fires [CellStateChanged](#) event. To change the radio cell state you have to call [CellState](#) property. The value for radio group number is not important. You can allocate any number that you want. To add or remove a cell to a given radio group you have to use CellRadioGroup property. By default, when a cell of radio type is created the radio cell is not grouped to any of existent radio groups.

The following sample shows how to set the radio type for all cells of the first column in the first group, and group all of them in the same radio group (1234):

```
Dim h As Variant
With ExplorerTree1.Groups(0)
    .BeginUpdate
    With .Items
        For Each h In ExplorerTree1.Groups(0).Items
            .CellHasRadioButton(h, 0) = True
            .CellRadioGroup(h, 0) = 1234
        Next
    End With
    .EndUpdate
End With
```

or

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal
Item As EXPLOREERTREELibCtl.HITEM)
    With Group.Items
        .CellHasRadioButton(Item, 0) = True
    End With
End Sub
```

```
.CellRadioGroup(Item, 0) = 1234
```

```
End With
```

```
End Sub
```

To find out the radio cell that is checked in the radio group 1234 you have to call: [MsgBox](#)
Group.Items.CellCaption(, Group.Items.CellChecked(1234))

The following sample group all cells of the first column into a radio group, and display the cell's checked on the radio group when the state of a radio group has been changed:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal  
Item As EXPLOREERTREELibCtl.HITEM)
```

```
With Group.Items
```

```
.CellHasRadioButton(Item, 0) = True
```

```
.CellRadioGroup(Item, 0) = 1234
```

```
End With
```

```
End Sub
```

```
Private Sub ExplorerTree1_CellStateChanged(ByVal Group As EXPLOREERTREELibCtl.IGroup,  
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)
```

```
Debug.Print "In the 1234 radio group the "" & Group.Items.CellCaption(  
Group.Items.CellChecked(1234)) & "" is checked."
```

```
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
```

```
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellSingleLine([Item as Variant], [ColIndex as Variant]) as CellSingleLineEnum

Retrieves or sets a value indicating whether the cell's caption is painted using one or more lines.

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.
CellSingleLineEnum	A CellSingleLineEnum expression that indicates whether the cell displays its caption using one or more lines.

By default the cell uses only a line to display its caption. Use the `
` HTML tag inside the [CellCaption](#) to break a line. When the CellSingleLine is False, the height of the item is computed based on each cell caption.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellState([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets the cell's state. Has effect only for check and radio cells.

Type	Description
Item as Variant	A long expression that indicates the item's handle that indicates the owner of the cell.
ColIndex as Variant	A long expression that identifies the column's index, or a string expression that specifies the column's caption or the column's key.
Long	A long value that indicates the cell's state.

The CellState property has effect only for check and radio cells. When the cell's state is changed the control fires the [CellStateChanged](#) event. Use the [FilterType](#) property on exCheck to filter for checked or unchecked items.

The following sample shows how to change the state for a cell to checked state:

```
Group.Items.CellState(Group.Items(0), 0) = 1,
```

The following sample shows how to change the state for a cell to unchecked state:

```
Group.Items.CellState(Group.Items(0), 0) = 0,
```

The following sample shows how to change the state for a cell to partial checked state:

```
Group.Items.CellState(Group.Items(0), 0) = 2
```

The following sample displays a message when a cell of radio or check type has changed its state:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal  
Item As EXPLOREERTREELibCtl.HITEM)  
    With Group.Items  
        .CellHasCheckBox(Item, 0) = True  
    End With  
End Sub  
  
Private Sub ExplorerTree1_CellStateChanged(ByVal Group As EXPLOREERTREELibCtl.IGroup,  
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)  
    Debug.Print "The cell """" & Group.Items.CellCaption(Item, ColIndex) & """" has changed  
its state. The new state is " & IIf(Group.Items.CellState(Item, ColIndex) = 0, "Unchecked",  
"Checked")  
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
```

```
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellStrikeOut([Item as Variant], [ColIndex as Variant]) as Boolean

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

The **CellStrikeOut** property specifies whether the cell's caption should appear in **strikeout**. To change the **strike out** attribute for the whole item call [ItemStrikeOut](#) property.

Note: A cell is the intersection of an item with a column. All properties that has an **Item** and a **ColIndex** parameters are referring to a cell. The **Item** parameter represents the handle of an item, and the **ColIndex** parameter indicates an index (a numerical value, see **Column.Index** property) of a column , the column's caption (a string value, see **Column.Caption** property), or a handle to a cell. Here's few hints how to use properties with **Item** and **ColIndex** parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True  
Group.Items.CellBold(Group.Items(0), 0) = True  
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

property Items.CellToolTip([Item as Variant], [ColIndex as Variant]) as String

Retrieves or sets a text that is used to show the tooltip's cell.

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 caption or the column's key.
String	A string expression that indicates the cell's tooltip.

By default, the CellToolTip property is "...". 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. The control fires the [ToolTip](#) event when the column's tooltip is about to be displayed. The [ToolTipPopDelay](#) property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the [ToolTipWidth](#) property to specify the width of the tooltip window. Use the [Background\(exToolTipAppearance\)](#) property indicates the visual appearance of the borders of the tooltips. Use the [Background\(exToolTipBackColor\)](#) property indicates the tooltip's background color. Use the [Background\(exToolTipForeColor\)](#) property indicates the tooltip's foreground color. Use the [ToolTip](#) property to specify a tooltip to be shown when the cursor hovers the group's caption. Use the [ShowToolTip](#) method to programmatically display a custom tooltip.

The tooltip supports the following HTML tags:

- ** ... ** displays the text in **bold**
- **<i> ... </i>** 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.

The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once


the user clicks/collapses/expands the caption.


- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** 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=rrgbb> ... </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=rrgbb> ... </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=rrgbb> ... </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=rrgbb> ... </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

- **number[:width]** 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.
- **key[:width]** 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 **&**; (&), **<**; (<), **>**; (>), **"**; (") 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 FFFFFFF;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. Almost all properties of [Items](#) object 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's caption should appear in underline.

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.
Boolean	A boolean expression that indicates whether the cell is underlined.

To change the underline attribute for the whole item call ItemUnderline property.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True  
Group.Items.CellBold(Group.Items(0), 0) = True  
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

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 identifies the item's handle
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.
VAlignmentEnum	A VAlignmentEnum expression that indicates the cell's vertically alignment.

Use the CellVAlignment property to specify the vertically alignment for the cell's caption. Use the [CellSingleLine](#) property to specify whether a cell uses single or multiple lines. Use the [CellHAlignment](#) property to align horizontally the cell. The +/- button is aligned accordingly to the cell's caption. Use the [Def\(exCellVAlignment\)](#) property to specify the same vertical alignment for the entire column.

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.

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.

The following sample splits the first visible cell in three cells:

```
With Tree1
  .BeginUpdate
  .DrawGridLines = exAllLines
  With .Items
    Dim h As HITEM, f As HCELL
    h = .FirstVisibleItem
    f = .ItemCell(h, 0)
    f = .SplitCell(, f)
```

```
.CellCaption(, f) = "Split 1"  
f = .SplitCell(, f)  
.CellCaption(, f) = "Split 2"  
End With  
.EndUpdate  
End With
```

The following sample specifies that the inner cell should have 32 pixels:

```
With Tree1  
.BeginUpdate  
.DrawGridLines = exAllLines  
With .Items  
Dim h As HITEM, f As HCELL  
h = .FirstVisibleItem  
f = .ItemCell(h, 0)  
f = .SplitCell(, f)  
.CellCaption(, f) = "Split"  
.CellWidth(, f) = 32  
End With  
.EndUpdate  
End With
```

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 checks whether an item has child items. Use the [ItemChild](#) property to get the first child item, if there is one, 0 else.

method Items.ClearCellBackColor ([Item as Variant], [ColIndex as Variant])

Clears 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 column's index or the cell's handle, a string expression that indicates the column's caption.

The ClearCellBackColor method clears the cell's background color when the [CellBackColor](#) property is used.

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 the cell's handle, a string expression that indicates the column's caption.

The ClearCellForeColor method clears the cell's foreground color when [CellForeColor](#) property was used.

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 <code>HITEM</code>	A long expression that indicates the item's handle.

The `ClearItemBackColor` method clears the item's background color when [ItemBackColor](#) property was used.

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 is used.

property Items.DefaultItem as HITEM

Retrieves or sets the default item's handle.

Type	Description
HITEM	A long expression that indicates the handle of the item that's used by all properties of the 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 greater than 65000.

So, if you pass zero to a property that has a parameter titled Item, the group takes instead the DefaultItem value.

method `Items.Edit` ([Item as Variant], [ColIndex as Variant])

Edits a 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.

The `Edit` method starts edit operation. The edit operation starts only if the group's [AllowEdit](#) property is `True`. When the edit operation starts the control fires the [BeforeCellEdit](#) event. Use the `BeforeCellEdit` event to cancel the edit operation. When the edit operation ends the control fires the [AfterCellEdit](#) event. Use the `AfterCellEdit` event to change the cell's caption after edit operation ends. The following snippet of code shows how to start editing the first cell: `Group.Items.Edit Group.Items(0), 0`.

The following sample shows how to change the cell's caption when the edit operation ends.

```
Private Sub ExplorerTree1_AfterCellEdit(ByVal Group As EXPLORERTREELibCtl.IGroup,
ByVal Item As EXPLORERTREELibCtl.HITEM, ByVal ColIndex As Long, ByVal NewCaption As
String)
    Group.Items.CellCaption(Item, ColIndex) = NewCaption
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an `Item` and a `ColIndex` parameters are referring to a cell. The `Item` parameter represents the handle of an item, and the `ColIndex` parameter indicates an index (a numerical value, see `Column.Index` property) of a column , the column's caption (a string value, see `Column.Caption` property), or a handle to a cell. Here's few hints how to use properties with `Item` and `ColIndex` parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
Group.Items.CellBold(Group.Items(0), 0) = True
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

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

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 the given item is in the visible client area.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that fits the client area.

The method doesn't expand parent items. The `EnsureVisibleItem` method scrolls the control's content until the item is visible. Use the [IsItemVisible](#) to check if an item fits the control's client area.

The following sample shows how to make visible the first item in the group:
`Group.Items.EnsureVisibleItem Group.Items(0)`

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. Before expanding/collapsing an item, the control fires the [BeforeExpandItem](#) event. Use the `BeforeExpandItem` to cancel expanding/collapsing of an item. After item was expanded/collapsed the control fires the [AfterExpandItem](#) event. The following samples shows how to expand the selected item:

`Group.Items.ExpandItem(Group.Items.SelectedItem()) = True`. The property has no effect if the item has no child items. To check if the item has child items you can use [ChildCount](#) property.

The following sample shows how to programmatically expand the selected item:

```
Private Sub ExplorerTree1_SelectionChanged(ByVal Group As  
EXPLORERTREELibCtl.IGroup)  
    With Group.Items  
        .ExpandItem(SelectedItem()) = True  
    End With  
End Sub
```

property Items.FindItem (Caption as Variant, [ColIndex as Variant], [StartIndex as Variant]) as HITEM

Finds an item, looking for Caption in ColIndex column. The searching starts at StartIndex item.

Type	Description
Caption as Variant	A Variant expression that indicates the caption that is searched for.
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.
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 group's item that matches [CellCaption](#)(Item, ColIndex) = Caption. The searching starts from item index Start Index. The searching is case sensitive only if the ASCIIUpper property is empty. The following sample shows how to select the first item that matches "DUMON" on the first column:

```
Group.Items.SelectItem(Group.Items.FindItem("DUMON", 0)) = True
```

property Items.FindItemData (UserData as Variant, [StartIndex as Variant]) as HITEM

Finds the item giving its data.

Type	Description
UserData as Variant	A Variant expression that indicates the value being searched.
StartIndex as Variant	A long expression that indicates the index 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.

The following sample selects the item based on its path:

```
Group.Items.SelectItem(Group.Items.FindPath("Files and Folders\Hidden Files and Folders\Do not show hidden files and folder")) = True
```

property Items.FirstVisibleItem as HITEM

Retrieves the handle of the first visible item into control.

Type	Description
HITEM	A long expression that indicates the handle of 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 group's client area.

The following sample enumerates the items that fit the group's client area:

```
On Error Resume Next
Dim h As HITEM
Dim i As Long, j As Long, nCols As Long
With ExplorerTree1.Groups(0)
    nCols = .Columns.Count
    With .Items
        h = .FirstVisibleItem
        While Not (h = 0) And .IsItemVisible(h)
            Dim s As String
            s = ""
            For j = 0 To nCols - 1
                s = s + .CellCaption(h, j) + Chr(9)
            Next
            Debug.Print s
            h = .NextVisibleItem(h)
        Wend
    End With
End With
```

property Items.FocusItem as HITEM

Retrieves the handle of item that has the focus.

Type	Description
HITEM	A long expression that indicates the handle of 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.

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 [CellCaption](#) property indicates the cell's caption.

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

In other words, all cells applies the format of the [FormatColumn](#) property, excepts the cells with the FormatCell property being set. If the cell belongs to a column with the [FireFormatColumn](#) property on True, the Value parameter of the [FormatColumn](#) event shows the newly caption for the cell to be shown.

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>' : '') + 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):

Name	A	B	C	A+B+C
Root				
Child 1	7+	3+	1=	\$11.00
Child 2	2+	6+	12=	\$19.00
Child 3	2+	2+	4=	\$8.00
Child 4	2+	9+	4=	\$15.00

The **value** keyword in the FormatColumn property indicates the value to be formatted.

The supported binary arithmetic operators are:

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

The supported unary boolean operators are:

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

The supported binary boolean operators are:

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

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

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

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

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

"expression ? true_part : false_part"

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

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

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

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

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

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

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

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

- **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 than using *if* and *or* expressions.

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

The supported conversion unary operators are:

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

Here's few predefined types:

- 0 - empty (not initialized)
- 1 - null
- 2 - short
- 3 - long
- 4 - float
- 5 - double
- 6 - currency
- 7 - date
- 8 - string

- 9 - object
- 10 - error
- 11 - boolean
- 12 - variant
- 13 - any
- 14 - decimal
- 16 - char
- 17 - byte
- 18 - unsigned short
- 19 - unsigned long
- 20 - long on 64 bits
- 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format " displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

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

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- *Grouping* - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit

indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.

- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
 - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
 - 1 - Negative sign, number; for example, -1.1
 - 2 - Negative sign, space, number; for example, - 1.1
 - 3 - Number, negative sign; for example, 1.1-
 - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

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

Other known operators for dates are:

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

*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. You can use variables to avoid computing several times the same thing.*

property Items.FullPath (Item as HITEM) as String

Returns the fully qualified path of the referenced item in the group.

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 CellCaption property values of all its ancestors.

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 [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.

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 0 then the <code>InsertControlItem</code> property inserts the ActiveX control as a root item. If the <code>Parent</code> property is referring a locked item (ItemLocked property), the <code>InsertControlItem</code> property doesn't insert a new child ActiveX, instead insert the ActiveX control to the locked item that's specified by the <code>Parent</code> 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, if it is required. An empty string, if the control doesn't require a runtime license key.
Return	Description
HITEM	A long expression that indicates the handle of the newly created item.

The control supports ActiveX hosting, so you can insert any ActiveX component. The `ControlID` must be formatted in one of the following ways:

- A ProgID such as "Exontrol.ExplorerTree"
- A CLSID such as "{8E27C92B-1264-101C-8A2F-040224009C02}"
- A URL such as "https://www.exontrol.com"
- A reference to an Active document such as "c:\temp\myfile.doc", or "c:\temp\picture.gif"
- A fragment of HTML such as "MSHTML:<HTML><BODY>This is a line of text</BODY></HTML>"
- A fragment of XML

Once that an item of ActiveX type has been added you can get the OLE control created using the `ItemObject` property. To check if an item contains an ActiveX control you can use `ItemControlID` property. To change the height of an ActiveX item you have to use `ItemHeight` property. When the group contains at least an item of ActiveX type, it is recommend to set [ScrollBySingleLine](#) property of group to true. Events from contained components are fired

through to your program using the exact same model used in VB6 for components added at run time (See [ItemOleEvent](#) event, [OleEvent](#) and [OleEventParam](#)). For instance, when an ActiveX control fires an event, the control forwards that event to your container using ItemOleEvent event of the control.

The following sample adds dynamically an ExplorerTree ActiveX Control and a Microsoft Calendar Control:

```
With ExplorerTree1.Groups(0)
  Dim hExplorerTree As HITEM
  hExplorerTree = .Items.InsertControlItem(.Items(0), "Exontrol.ExplorerTree")
  .Items.ItemHeight(hExplorerTree) = 212
  With .Items.ItemObject(hExplorerTree)
    With .Groups.Add("Inside Group")
      With .Items
        .AddItem "One"
        .AddItem "Two"
        .AddItem "Three"
      End With
    End With
  End With
End With

Dim hCalc As HITEM
hCalc = .Items.InsertControlItem( "MSCal.Calendar")
With .Items.ItemObject(hCalc)
  .ShowTitle = False
  .ShowDateSelectors = False
End With
End With
```

The following sample shows how to handle any event that a contained ActiveX fires:

```
Private Sub ExplorerTree1_ItemOleEvent(ByVal Group As EXPLORERTREELibCtl.IGroup,
ByVal Item As EXPLORERTREELibCtl.HITEM, ByVal Ev As EXPLORERTREELibCtl.IOleEvent)
  On Error Resume Next
  Dim i As Long
  Debug.Print "The " & Ev.Name & " was fired. "
  If Not (Ev.CountParam = 0) Then
    Debug.Print "The event has the following parameters: "
```

```

For i = 0 To Ev.CountParam - 1
    Debug.Print " - " & Ev(i).Name & " = " & Ev(i).Value
Next
End If
End Sub

```

Some of ActiveX controls requires additional window styles to be added to the container window. For instance, the Web Browser added by the Group.Items.InsertControlItem(["https://www.exontrol.com"](https://www.exontrol.com)) won't add scroll bars, so you have to do the following:

First thing is to declare the WS_HSCROLL and WS_VSCROLL constants at the top of your module:

```

Private Const WS_VSCROLL = &H200000
Private Const WS_HSCROLL = &H100000

```

Then you need to to insert a Web control use the following lines:

```

Dim hWeb As HITEM
hWeb = Group.Items.InsertControlItem("https://www.exontrol.com")
Group.Items.ItemHeight(hWeb) = 196

```

Next step is adding the AddItem event handler:

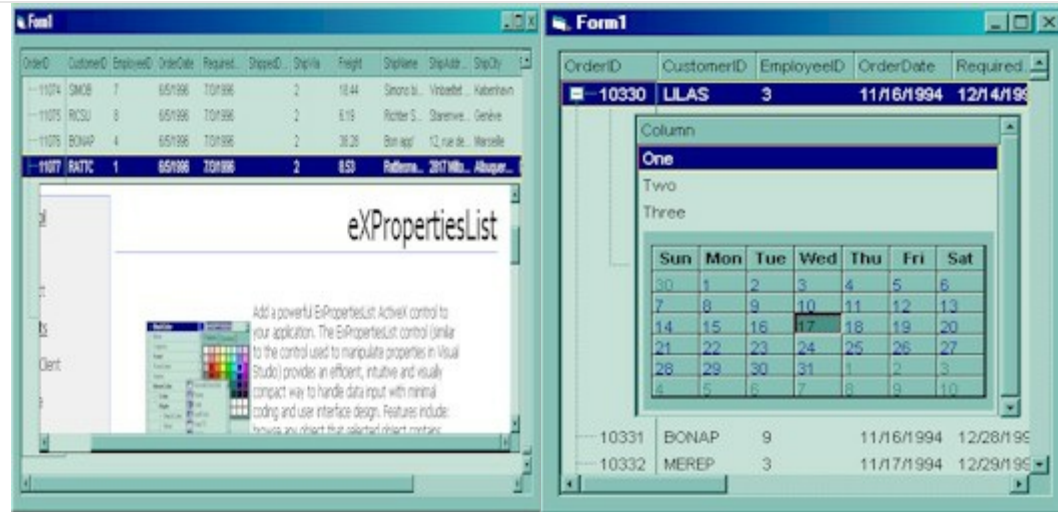
```

Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal
Item As EXPLORERTREELibCtl.HITEM)
    If (Group.Items.ItemControlID(Item) = "https://www.exontrol.com") Then
        ' Some of controls like the WEB control, requires some additional window styles ( like
WS_HSCROLL and WS_VSCROLL window styles )
        ' for the window that host that WEB control, to allow scrolling the web page
        Group.Items.ItemWindowHostCreateStyle(Item) =
Group.Items.ItemWindowHostCreateStyle(Item) + WS_HSCROLL + WS_VSCROLL
    End If
End Sub

```

If somehow the InsertItemControl wasn't able to create your ActiveX on some Windows platforms, and you don't know why, you can use the following code to make sure that ActiveX control can be created properly by using (the sample is trying to add a new Microsoft RichText ActivX control into your form):

Controls.Add "RICHTEXT.RichtextCtrl", "rich"



method Items.InsertItem ([Parent as HITEM], [UserData as Variant], [Caption 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.
Caption as Variant	A string expression that indicates the cell's caption on the first column, a safe array that holds the caption for each column.

Return	Description
HITEM	Retrieves the handle of the newly created item.

The InsertItem property fires the [AddItem](#) event. Use the InsertItem property to add a new child to an item. 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 [CellCaptionFormat](#) property to specify whether the cell displays the caption using the HTML format. Use the [CellCaption](#) property to assign captions for cells in a multi-column group. Use the [LockedItemCount](#) property to lock or unlock items to the top or bottom side of the group. Use the [MergeCells](#) method to combine two or more cells in a single cell. Use the [SplitCell](#) property to split a cell.

The following sample shows how to create a simple hierarchy into your group:

```
With ExplorerTree1.Groups.Add("Simple Tree")
```

```
.AutoHeight = True
```

```
.ColumnAutoResize = True
```

```
.LinesAtRoot = True
```

```
.FullRowSelect = False
```

```
.MarkSearchColumn = False
```

```
.BeginUpdate
```

```
    ' By default the group adds a default column
```

```
    With .Items
```

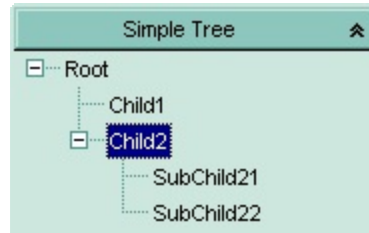
```
        Dim h As HITEM
```

```

h = .InsertItem(, , "Root")
.InsertItem h, , "Child1"
h = .InsertItem(h, , "Child2")
.InsertItem h, , "SubChild21"
h = .InsertItem(h, , "SubChild22")
End With
.EndUpdate

```

End With



The following VB sample adds items when control has multiple columns:

```

With ExplorerTree1.Groups.Add("Simple Tree")

```

```

.AutoHeight = True
.ColumnAutoSize = False
.LinesAtRoot = exLinesAtRoot
.HeaderVisible = True

```

```

.BeginUpdate

```

' By default the group adds a default column, so we only change it's caption

```

With .Columns(0)

```

```

.Caption = "Column 1"

```

```

.Width = 64

```

```

End With

```

```

.Columns.Add "Column 2"

```

```

.Columns.Add "Column 3"

```

```

Dim h As HITEM

```

```

With .Items

```

```

h = .AddItem(Array("Item 1.1", "Item 1.2", "Item 1.3"))

```

```

.InsertItem h, , Array("Item 2.1", "Item 2.2", "Item 2.3")

```

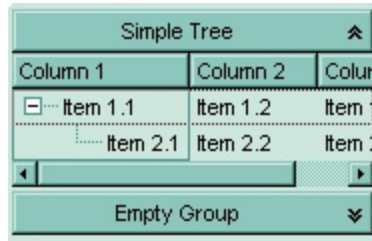
End With

.EndUpdate

End With

With ExplorerTree1.Groups.Add("Empty Group")

End With



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) as Boolean

Checks if the specific item fits the group's client area.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that fits the client area.
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 group's client area.

The following sample enumerates the items that fit the group's client area:

```
On Error Resume Next
Dim h As HITEM
Dim i As Long, j As Long, nCols As Long
With ExplorerTree1.Groups(0)
    nCols = .Columns.Count
    With .Items
        h = .FirstVisibleItem
        While Not (h = 0) And .IsItemVisible(h)
            Dim s As String
            s = ""
            For j = 0 To nCols - 1
                s = s + .CellCaption(h, j) + Chr(9)
            Next
            Debug.Print s
            h = .NextVisibleItem(h)
        Wend
    End With
End With
```


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 when the item hosts an ActiveX control.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that was previously created by InsertControlItem property.
AppearanceEnum	An AppearanceEnum expression that indicates the item's appearance.

Use the ItemAppearance property to specify the item's appearance if the item is of ActiveX type.

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 handle of the item.
Color	A color expression that indicates the item's background color.

To change the background color for a cell you can call [CellBackColor](#). To change the background color of the entire control you can call [BackColor](#) property of the control.

The following sample changes the background color for cells in the first column:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
    With Group.Items  
        .CellBackColor(Item, 0) = vbBlue  
    End With  
End Sub
```

property Items.ItemBold(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in bold.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Boolean	A boolean expression that indicates whether the item should appear in bold.

To change the bold attribute for a cell you can call [CellBold](#) property.

The following sample bolds the selected item:

```
Dim hOldBold As HITEM
```

```
Private Sub ExplorerTree1_SelectionChanged(ByVal Group As  
EXPLORERTREELibCtl.IGroup)  
    If (Group.Index = 0) Then  
        With Group.Items  
            If Not hOldItem = 0 Then  
                .ItemBold(hOldItem) = False  
            End If  
            hOldItem = .SelectedItem()  
            .ItemBold(hOldItem) = True  
        End With  
    End If  
End Sub
```

property Items.ItemByIndex (Index as Long) as HITEM

Retrieves the handle of the item given its index in Items collection..

Type	Description
Index as Long	A long expression that indicates the index of the item.
HITEM	A long expression that indicates the item's handle.

Use the ItemByIndex to get the index of an item. Use the [ItemPosition](#) property to get the item's position. Use the [ItemToIndex](#) property to get the index of giving item.

The following statements are equivalents: Group.Items(0), Group.Items.ItemByIndex(0).

The following sample displays the handle for each item into Items collection:

```
Dim i As Long, n As Long
With Group.Items
    n = .ItemCount
    For i = 0 To n - 1
        Debug.Print .ItemByIndex(i)
    Next
End With
```

property Items.ItemCell (Item as HITEM, ColIndex as Variant) as HCELL

Retrieves the cell's handle based on a specific column.

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 or the cell's handle, a string expression that indicates the column's caption.
HCELL	A long expression that indicates the handle of the cell.

A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
Group.Items.CellBold(, Group.Items.ItemCell(Group.Items(0), 0)) = True
```

```
Group.Items.CellBold(Group.Items(0), 0) = True
```

```
Group.Items.CellBold(Group.Items(0), "ColumnName") = True
```

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 handle of the first child item.

If the ItemChild property gets 0, the item has no child items. Use this property to get the first child of an item. [NextVisibleItem](#) or [NextSiblingItem](#) to get the next visible, sibling item.

The following snippet of code shows how to recursively scan all child items in the group:

```
Sub Recltem(ByVal g As EXPLORERTREELibCtl.Group, ByVal h As  
EXPLORERTREELibCtl.HITEM)  
  If Not (h = 0) Then  
    Dim hChild As HITEM  
    With g.Items  
      Debug.Print .CellCaption(h, 0)  
      hChild = .ItemChild(h)  
      While Not (hChild = 0)  
        Recltem g, hChild  
        hChild = .NextSiblingItem(hChild)  
      Wend  
    End With  
  End If  
End Sub
```

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 that was previously created by the InsertControlItem property.
String	A string expression that indicates the control identifier used by InsertControlItem method 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 the Items collection.

The ItemCount property counts the items in the collection. Use [ChildCount](#) to get the number of child items.

The following sample enumerates all control items:

```
Dim i As Long, n As Long
With Group.Items
  n = .ItemCount
  For i = 0 To n - 1
    Debug.Print .ItemByIndex(i)
  Next
End With
```

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 item's handle that has associated some extra data.
Variant	A variant value that indicates the item's extra data.

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 group doesn't use this value.

property Items.ItemDivider(Item as HITEM) as Long


Specifies whether the item acts like a divider item. The value indicates the index of column used to define the divider's title.

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. The `ItemDivider` property specifies the index of the cell being displayed. In other words, the divider item merges the item cells into a single cell. Use the [ItemDividerLine](#) property to define the line that a divider item paints. Use the [LockedItemCount](#) property to lock items on the top or bottom side of the control.

The following sample adds a divider item that's locked to the top side of the first group in the control: (Before running this sample please make sure that your control contains groups objects):

```
With ExplorerTree1.Groups(0)
    .BeginUpdate
    .DrawGridLines = exNoLines
    With .Items
        .LockedItemCount(TopAlignment) = 1
        Dim h As HITEM
        h = .LockedItem(TopAlignment, 0)
        .ItemDivider(h) = 0
        .ItemHeight(h) = 24
        .CellCaption(h, 0) = "<b>Total</b>: $12.344.233"
        .CellCaptionFormat(h, 0) = exHTML
        .CellHAlignment(h, 0) = RightAlignment
    End With
    .EndUpdate
End With
```

Group 1				⬆
	new Column			⬆
		new Column		
	Column 1		new Column	
this is a divider item fixed to the top side of the group's list				
<input type="checkbox"/> Root 1 . Just an item that merges four cells.				
	Child 1	Subitem 2	Subitem 3	Subitem 4
	Child 2	Subitem 2	Subitem 3	Subitem 4
Group 2				⬆
				Tip Total: \$2,000,000
<hr/>				
<input type="checkbox"/> Root 1				
				Grand Total: \$2,000,000

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.

Use the [ItemDivider](#) property to define a divider item. Use the [ItemDividerLine](#) and [ItemDividerAlignment](#) properties to define the style of the line into a divider item.

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.

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.ItemFont (Item as HITEM) as IFontDisp

Retrieves or sets the item's font.

Type	Description
Item as HITEM	A long expression that specifies the item's handle.
IFontDisp	A Font object that specifies the item's font.

By default, the ItemFont property is nothing. If the ItemFont property is nothing, the item uses the group's [Font](#), and the ItemFont property is nothing. 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 [CellCaptionFormat](#) to specify different font attributes.

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.

To change the foreground color for a cell call [CellForeColor](#) property. To change the foreground color for the entire control call [ForeColor](#) property of the control.

The following sample to change the foreground color for the first column:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
    With Group.Items  
        .CellForeColor(Item, 0) = vbBlue  
    End With  
End Sub
```


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.

Use the ItemHasChildren property to build a virtual tree. Use the [BeforeExpandItem](#) event to add new child items before expanding a fake item. Use the [ItemChild](#) property to get the first child item. Use the ItemChild or [ChildCount](#) property to determine whether an item contains child items.

```
Private Sub ExplorerTree1_BeforeExpandItem(ByVal Group As EXPLORERTREELibCtl.IGroup,
ByVal Item As EXPLORERTREELibCtl.HITEM, Cancel As Variant)
    With Group.Items
        If (.ItemData(Item) = 1234) Then
            add (Item)
            .ItemData(Item) = 1111
        End If
    End With
End Sub
```

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 item's handle.
Long	A long value that indicates the item's height in pixels.

To change the default height of the item before inserting items to collection you can call [DefaultItemHeight](#) property of the group. 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. Also, please check the [CellSingleLine](#) property.

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 that uses italic font attribute.
Boolean	A boolean expression that indicates whether the item should appear in italic.

To change the italic attribute for a cell you can call [CellItalic](#) property.

The following sample applies an italic font attribute to the selected item in the first group:

```
Private Sub ExplorerTree1_SelectionChanged(ByVal Group As  
EXPLORERTREELibCtl.IGroup)  
    If (Group.Index = 0) Then  
        With Group.Items  
            If Not hOldItem = 0 Then  
                .ItemItalic(hOldItem) = False  
            End If  
            hOldItem = .SelectedItem()  
            .ItemItalic(hOldItem) = True  
        End With  
    End If  
End Sub
```

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.
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.

property Items.ItemObject (Item as HITEM) as Object

Retrieves the item's ActiveX object associated, if the item was previously created by InsertControllItem property.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that was previously created by InsertControllItem property.
Object	An object that indicates the ActiveX hosted by the item.

Use the ItemObject to access item's ActiveX properties and methods, if the item was created by the [InsertControllItem](#) property.

The following sample adds an ExplorerTree ActiveX control:

```
With ExplorerTree1.Groups.Add("ActiveX")
  Dim hExplorerTree As HITEM
  hExplorerTree = .Items.InsertControllItem(.Items(0), "Exontrol.ExplorerTree")
  .Items.ItemHeight(hExplorerTree) = 212
  With .Items.ItemObject(hExplorerTree)
    With .Groups.Add("Inside Group")
      With .Items
        .AddItem "One"
        .AddItem "Two"
        .AddItem "Three"
      End With
    End With
  End With
End With
```

property Items.ItemParent (Item as HITEM) as HITEM

Returns the handle of the parent 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 parent item.

To change the item's parent call [SetParent](#) method. 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 group is able to handle more root items. To get the collection of root items you can use [RootCount](#) and [RootItem](#) properties.

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. You can use the ItemPosition property to change the item's position after it been added to collection. When the group sorts the tree, the item for each position can be changed, so you can use the item's handle or item's index to identify an item.

property Items.ItemStrikeOut(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in strikeout.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the item should appear in strikeout.

To change the strike out attribute for a cell you can call [CellStrikeOut](#) property.

The following sample applies a strikeout font attribute for the selected item in the first group:

```
Private Sub ExplorerTree1_SelectionChanged(ByVal Group As  
EXPLORERTREELibCtl.IGroup)  
    If (Group.Index = 0) Then  
        With Group.Items  
            If Not hOldItem = 0 Then  
                .ItemStrikeOut(hOldItem) = False  
            End If  
            hOldItem = .SelectedItem()  
            .ItemStrikeOut(hOldItem) = True  
        End With  
    End If  
End Sub
```


property Items.ItemToIndex (Item as HITEM) as Long

Retrieves the index of item into 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 the item in Items collection.

Use the ItemToIndex property to get the item's index into Items collection. Use [ItemPosition](#) property to change the item's position. Use the [ItemByIndex](#) property to get an item giving its index.

property Items.ItemUnderline(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in underline.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the item should appear in underline.

To change the underline attribute for a cell you can call [CellUnderline](#) property.

The following sample underlines the selected item in the first group:

```
Private Sub ExplorerTree1_SelectionChanged(ByVal Group As  
EXPLORERTREELibCtl.IGroup)  
    If (Group.Index = 0) Then  
        With Group.Items  
            If Not hOldItem = 0 Then  
                .ItemUnderline(hOldItem) = False  
            End If  
            hOldItem = .SelectedItem()  
            .ItemUnderline(hOldItem) = True  
        End With  
    End If  
End Sub
```

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.
Long	A long expression that indicates the item's width, when the item contains an ActiveX control.

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.

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 $-(\text{ItemWidth}+32000)$ 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](#) method.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that was previously created by InsertControlItem method.
Long	A long value that indicates the window handle that hosts the item's ActiveX.

The ItemWindowHost property retrieves the window's handle that is the container for the item's ActiveX control. Use the [ItemObject](#) property to access the ActiveX properties and methods.

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 handle of the item that was previously created by InsertControlItem method.
Long	A long value that indicates the container window's style.

The ItemWindowHostCreateStyle property has no effect for regular items. ItemWindowHostCreateStyle property must be change in the [AddItem](#) event.

Some of ActiveX controls requires additional window styles to be added to the conatiner window. For instance, the Web Brower added by the Group.Items.InsertControlItem(["https://www.exontrol.com"](https://www.exontrol.com)) won't add scroll bars, so you have to do the following:

First thing is to declare the WS_HSCROLL and WS_VSCROLL constants at the top of your module:

```
Private Const WS_VSCROLL = &H200000  
Private Const WS_HSCROLL = &H100000
```

Then you need to to insert a Web control use the following lines:

```
Dim hWeb As HITEM  
hWeb = Group.Items.InsertControlItem("https://www.exontrol.com")  
Group.Items.ItemHeight(hWeb) = 196
```

Next step is adding the AddItem event handler:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
    If (Group.Items.ItemControlID(Item) = "https://www.exontrol.com") Then  
        ' Some of controls like the WEB control, requires some additional window styles ( like  
WS_HSCROLL and WS_VSCROLL window styles )  
        ' for the window that host that WEB control, to allow scrolling the web page  
        Group.Items.ItemWindowHostCreateStyle(Item) =  
Group.Items.ItemWindowHostCreateStyle(Item) + WS_HSCROLL + WS_VSCROLL  
    End If  
End Sub
```

4 Fund


OrderID	CustomerID	EmployeeID	OrderDate	Required	ShippedID	ShipVia	Freight	ShipName	ShipAddr	ShipCity
11074	SMCB	7	6/5/96	7/5/96	2	18.44	Simon S.	Vindicta	Karenham	
11075	RCSU	8	6/5/96	7/5/96	2	6.18	Richter S.	Starview	Genève	
11076	BNWP	4	6/5/96	7/5/96	2	38.26	Bon app	12 rue de	Wavrele	
11077	BATC	1	6/5/96	7/5/96	2	6.53	Pelloux	2017 Wb.	Albuquerque	

eXPropertiesList

Client

Add a powerful EPropertiesList ActiveX control to your application. The EPropertiesList control (similar to the control used to manipulate properties in Visual Studio) provides an efficient, intuitive and visually compact way to handle data input with minimal coding and user interface design. Features include:

- Focus on object that selected object contains



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 handle of the last visible item.

To get the first visible item use [FirstVisibleItem](#) property. The LastVisibleItem property retrieves the handle for the last 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 group's client area.

The following sample enumerates the items that fit the group's client area:

```
On Error Resume Next
Dim h As HITEM
Dim i As Long, j As Long, nCols As Long
With ExplorerTree1.Groups(0)
    nCols = .Columns.Count
    With .Items
        h = .FirstVisibleItem
        While Not (h = 0) And .IsItemVisible(h)
            Dim s As String
            s = ""
            For j = 0 To nCols - 1
                s = s + .CellCaption(h, j) + Chr(9)
            Next
            Debug.Print s
            h = .NextVisibleItem(h)
        Wend
    End With
End With
```

property Items.LockedItem (Alignment as VAlignmentEnum, Index as Long) as HITEM

Retrieves the handle of the locked/fixed 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

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 LockedItem property to access a locked item by its position. Use the [LockedItemCount](#) property to add or remove items fixed/locked to the top or bottom side of the control. Use the [ShowLockedItems](#) property to show or hide the locked items. Use the [IsItemLocked](#) property to check whether an item is locked or unlocked. Use the [CellCaption](#) property to specify the caption for a cell. Use the [InsertControlItem](#) property to assign an ActiveX control to a locked item only.

The following sample adds a divider item that's locked to the top side of the first group in the control: (Before running this sample please make sure that your control contains groups objects):

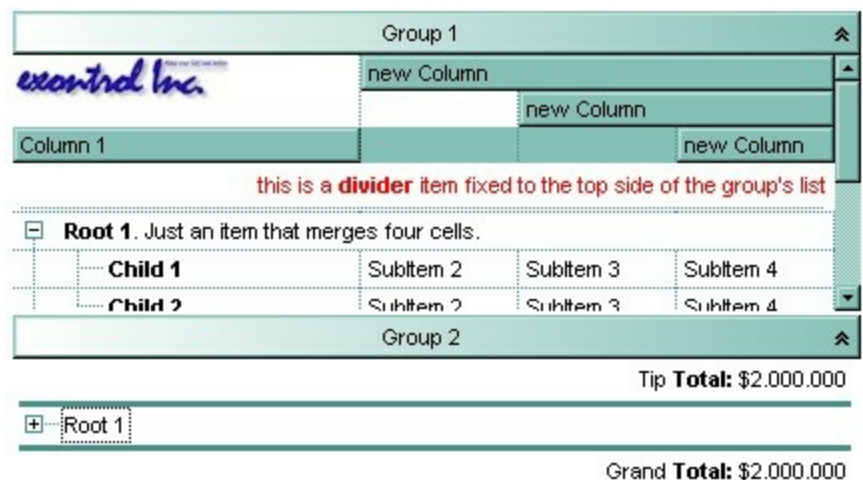
```
With ExplorerTree1.Groups(0)
    .BeginUpdate
    .DrawGridLines = exNoLines
    With .Items
        .LockedItemCount(TopAlignment) = 1
        Dim h As HITEM
        h = .LockedItem(TopAlignment, 0)
        .ItemDivider(h) = 0
        .ItemHeight(h) = 24
        .CellCaption(h, 0) = "<b>Total</b>: $12.344.233"
        .CellCaptionFormat(h, 0) = exHTML
        .CellHAlignment(h, 0) = RightAlignment
    End With
    .EndUpdate
```


property Items.LockedItemCount(Alignment as VAlignmentEnum) as Long

Specifies the number of items fixed on the top or bottom side of the group.

Type	Description
Alignment as VAlignmentEnum	A VAlignmentEnum expression that specifies the top or bottom side of the group.
Long	A long expression that indicates the number of items locked to the top or bottom side of the group.

A locked or fixed item is always displayed on the top or bottom side of the group no matter if the group'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 group. 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 [CellCaption](#) property to specify the caption for a cell. Use the [CountLockedColumns](#) property to lock or unlock columns in the group. Use the [ItemBackColor](#) property to specify the item's background color. Use the [ItemDivider](#) property to merge the cells.



The following sample adds a divider item that's locked to the top side of the first group in the control: (Before running this sample please make sure that your control contains groups objects):

```
With ExplorerTree1.Groups(0)
    .BeginUpdate
    .DrawGridLines = exNoLines
With .Items
    .LockedItemCount(TopAlignment) = 1
    Dim h As HITEM
    h = .LockedItem(TopAlignment, 0)
```

.ItemDivider(h) = 0

.ItemHeight(h) = 24

.CellCaption(h, 0) = "Total: \$12.344.233"

.CellCaptionFormat(h, 0) = exHTML

.CellHAlignment(h, 0) = RightAlignment

End With

.EndUpdate

End With

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 retrieve 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 retrieve 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 [CellCaption](#) property to specify the cell's caption. Use the [ItemCell](#) property to retrieve 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.

The following sample adds three columns to a group, a root item and few child items:

```
With ExplorerTree1
  .BeginUpdate
  With .Groups.Add("Group")
    .BeginUpdate
    .Expanded = True
    .AutoHeight = True
    .MarkSearchColumn = False
    .DrawGridLines = exAllLines
```

```
.LinesAtRoot = exLinesAtRoot
With .Columns.Item(0)
    .Def(exCellCaptionFormat) = exHTML
    .Width = 64
```

```
End With
```

```
.Columns.Add "Column 2"
```

```
.Columns.Add "Column 3"
```

```
.ColumnAutoSize = True
```

```
With .Items
```

```
    Dim h As Long
```

```
    h = .AddItem("Root. This is the root item")
```

```
    .InsertItem h, , Array("Child 1", "SubItem 2", "SubItem 3")
```

```
    .InsertItem h, , Array("Child 2", "SubItem 2", "SubItem 3")
```

```
    .ExpandItem(h) = True
```

```
End With
```

```
.EndUpdate
```

```
End With
```

```
.EndUpdate
```

```
End With
```

Group		
[-] Root. This is th		
[-] Child 1	SubItem 2	SubItem 3
[-] Child 2	SubItem 2	SubItem 3

(Notice that the caption of the root item is truncated by the column that belongs to).

If we are merging the first three cells in the root item we get:

Group		
[-] Root. This is the root item		
[-] Child 1	SubItem 2	SubItem 3
[-] Child 2	SubItem 2	SubItem 3

You can merge the first three cells in the root item using any of the following methods:

```
With ExplorerTree1.Groups(0)
```

```
    With .Items
```

```
        .CellMerge(.RootItem(0), 0) = Array(1, 2)
```

```
    End With
```

```
End With
```

With ExplorerTree1.Groups(0)

.BeginUpdate

With .Items

Dim r As Long

r = .RootItem(0)

.CellMerge(r, 0) = 1

.CellMerge(r, 0) = 2

End With

.EndUpdate

End With

With ExplorerTree1.Groups(0)

.BeginUpdate

With .Items

Dim r As Long

r = .RootItem(0)

.MergeCells .ItemCell(r, 0), .ItemCell(r, 1)

.MergeCells .ItemCell(r, 0), .ItemCell(r, 2)

End With

.EndUpdate

End With

With ExplorerTree1.Groups(0)

With .Items

Dim r As Long

r = .RootItem(0)

.MergeCells .ItemCell(r, 0), **Array**(.ItemCell(r, 1), .ItemCell(r, 2))

End With

End With

With ExplorerTree1.Groups(0)

With .Items

Dim r As Long

r = .RootItem(0)

.MergeCells **Array**(.ItemCell(r, 0), .ItemCell(r, 1), .ItemCell(r, 2))

End With

End

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 handle of the next sibling item.

Use [ItemChild](#) and [NextSiblingItem](#) properties to scan the collection of child items.

The following sample shows how to scan recursively all child items hosted by an item:

```
Sub Recltem(ByVal g As EXPLORERTREELibCtl.Group, ByVal h As  
EXPLORERTREELibCtl.HITEM)  
    If Not (h = 0) Then  
        Dim hChild As HITEM  
        With g.Items  
            Debug.Print .CellCaption(h, 0)  
            hChild = .ItemChild(h)  
            While Not (hChild = 0)  
                Recltem g, hChild  
                hChild = .NextSiblingItem(hChild)  
            Wend  
        End With  
    End If  
End Sub
```


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 [IsItemVisible](#) property to check whether an item fits the control's client area.

The following sample enumerates the visible items:

```
Private Sub VisItems(ByVal g As EXPLORERTREELibCtl.Group)
    Dim h As HITEM
    With g.Items
        h = .FirstVisibleItem
        While Not (h = 0)
            Debug.Print .CellCaption(h, 0)
            h = .NextVisibleItem(h)
        Wend
    End With
End Sub
```

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 "\".

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. Also, please check the [NextSiblingItem](#) property.

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 more previous visible items. Also, please check the [NextVisibleItem](#) property.

method `Items.RemoveAllItems ()`

Removes all items from the control.

Type	Description
------	-------------

Use the [Clear](#) method of Columns object to erase the group's content. Use the `RemoveAllItems` to clear the Items collection. To remove an item from the Items collection use [RemoveItem](#) method.

method Items.RemoveItem (Item as HITEM)

Removes a specific item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being removed.

The following sample shows how to remove the first item in the group:

Group.Items.RemoveItem Group.Items(0). To remove all items in the collection you can use [RemoveAllItems](#) method. The RemoveItem method doesn't remove recursively the item.

The following removes recursively an item:

```
Private Sub RemoveItemRec(ByVal g As EXPLORERTREELibCtl.Group, ByVal h As  
EXPLORERTREELibCtl.HITEM)  
    With g.Items  
        g.BeginUpdate  
        If (.ChildCount(h) > 0) Then  
            Dim hChild As HITEM  
            hChild = .ItemChild(h)  
            While (hChild <> 0)  
                Dim hNext As HITEM  
                hNext = .NextSiblingItem(hChild)  
                RemoveItemRec g, hChild  
                hChild = hNext  
            Wend  
        End If  
        .RemoveItem h  
    End With  
End Sub
```

property Items.RootCount as Long

Retrieves the number of root objects into Items collection.

Type	Description
Long	A long value that indicates the count of root items in the 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.

The following sample enumerates all root items:

```
Dim i As Long, n As Long
With Group.Items
    n = .RootCount
    For i = 0 To n - 1
        Debug.Print .CellCaption(.RootItem(i), 0)
    Next
End With
```

Property Items.RootItem ([Position as Long]) as HITEM

Retrieves the handle of the root item giving its index into the root items collection.

Type	Description
Position as Long	A long value that indicates the position of the root item being accessed.
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.

The following sample enumerates all root items:

```
Dim i As Long, n As Long
With Group.Items
  n = .RootCount
  For i = 0 To n - 1
    Debug.Print .CellCaption(.RootItem(i), 0)
  Next
End With
```


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 [SelectionChanged](#) 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.

method `Items.SelectAll ()`

Selects all items.

Type	Description
------	-------------

property Items.SelectCount as Long

Counts the number of items that are selected into control.

Type	Description
Long	A long expression that identifies the number of selected items.

Each group supports multiple items selection. Use [SingleSel](#) property of the Group object to allow multiple selection. The SelectCount property counts the selected items in the group. If the control's SingleSel is False, the following statement retrieves the handle for the selected item: Group.Items.SelectedItem(). If the control supports multiple selection the following sample enumeratee all selected items:

```
Private Sub selItems(ByVal g As EXPLORERTREELibCtl.Group)
    Dim h As HITEM
    Dim i As Long, j As Long, nCols As Long, nSels As Long
    With g
        nCols = .Columns.Count
        With .Items
            nSels = .SelectCount
            For i = 0 To nSels - 1
                Dim s As String
                For j = 0 To nCols - 1
                    s = s + .CellCaption(.SelectedItem(i), j) + Chr(9)
                Next
                Debug.Print s
            Next
        End With
    End With
End Sub
```

property Items.SelectedItem ([Index as Long]) as HITEM

Retrieves the selected item's handle given its index in selected items collection.

Type	Description
Index as Long	Identifies the index of the selected item into the selected items collection.
HITEM	A long expression that indicates the handle of the selected item.

If the control support multiple selection, you can use the [SelectCount](#) property to find out how many items are selected in the control.

The following sample shows how to print the caption for the selected cell: `Debug.Print Group.Items.CellCaption(Group.Items.SelectedItem(0), 0)`.

The following sample underlines the selected item in the first group:

```
Private Sub ExplorerTree1_SelectionChanged(ByVal Group As  
EXPLORERTREELibCtl.IGroup)  
    If (Group.Index = 0) Then  
        With Group.Items  
            If Not hOldItem = 0 Then  
                .ItemUnderline(hOldItem) = False  
            End If  
            hOldItem = .SelectedItem()  
            .ItemUnderline(hOldItem) = True  
        End With  
    End If  
End Sub
```

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 that is selected or unselected.
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 [SelectedItem](#) property to get the selected items.

The following sample shows how to select the first created item:

```
Tree1.Items.SelectItem(Tree1.Items(0)) = True
```

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 handle of the item being moved.
NewParent as HITEM	A long expression that indicates the handle of the new parent item.

Use [AcceptSetParent](#) property to verify if the the parent of an item can be changed. The following sample shows how to set the parent of first item to second item:

`Group.Items.SetParent Group.Items(0), Group.Items(1)`. To retrieve the parent of a given item you can use `ItemParent` property

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. Though, 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. 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 group.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that is going to be sorted.
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.
Ascending as Boolean	A boolean expression that defines the sort order.

The `SortChildren` will not recurse through the tree, only the immediate children of item will be sorted. If your group looks like a simple list you can use the following line of code to sort ascending the list by first column: `Group.Items.SortChildren 0, 0`. 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 group automatically sorts the children items when user clicks on column's header. The [SortOrder](#) property sorts the items and displays the sorting icon in the column's header.

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 [CellCaption](#) property to assign a caption 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.

("Merge" means multiple cells in a single cell, "Split" means multiple cells **inside** a single cell)

Column 1	Column 2
Item	inner cell 1 inner cell 2 subitem 1
Root	
Child 1	
Child 2	

(the picture shows a cell that's divided in three cells)

The following sample shows how to split a cell in three cells:

```
bInner = True
```

With ExplorerTree1

.BeginUpdate

.BackColor = vbWhite

With .Groups.Add("Tree")

.BeginUpdate

.Expanded = True

.LinesAtRoot = exLinesAtRoot

.DrawGridLines = exAllLines

.MarkSearchColumn = False

.HeaderVisible = True

.Columns.Add("Column 2").Width = 256

With .Items

Dim h As HITEM, hCell As hCell

h = .AddItem(Array("Item", "subitem 1"))

If (blInner) Then

f = .SplitCell(h, 0)

.CellCaption(, f) = **inner** cell 1"

.CellCaptionFormat(, f) = exHTML

f = .SplitCell(, f)

.CellCaption(, f) = **inner** cell 2"

.CellCaptionFormat(, f) = exHTML

End If

h = .AddItem("Root ")

.CellMerge(h, 0) = 1

.InsertItem h, , "Child 1"

.InsertItem h, , "Child 2"

.ExpandItem(h) = True

End With

.EndUpdate

End With

.EndUpdate

End With

Column 1	Column 2
Item	subitem 1
Root	
Child 1	
Child 2	

(the picture shows a cell without being divided)

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.

The following sample shows few methods to unmerge cells:

```
With ExplorerTree1.Groups (0)
  With .Items
    .UnmergeCells .ItemCell(.RootItem(0), 0)
  End With
End With
```

```
With ExplorerTree1.Groups (0)
  With .Items
    Dim r As Long
    r = .RootItem(0)
    .UnmergeCells Array(.ItemCell(r, 0), .ItemCell(r, 1))
  End With
End With
```

```
With ExplorerTree1.Groups (0)
  .BeginUpdate
  With .Items
    .CellMerge(.RootItem(0), 0) = -1
    .CellMerge(.RootItem(0), 1) = -1
    .CellMerge(.RootItem(0), 2) = -1
  End With
  .EndUpdate
End With
```

method `Items.UnselectAll ()`

Unselects all items.

Type	Description
------	-------------

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 UnsplitCells 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 UnsplitCells 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).

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 group's client area.

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 list.

The VisibleItemCount property specifies the number of visible items in the list. For instance, you can use the VisibleItemCount property to specify the number of items being displayed after a filter is applied. 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

OleEvent object

The OleEvent object holds information about an event fired by an ActiveX control hosted by in item that was created using the [InsertControlItem](#) method.

Name	Description
CountParam	Retrieves the count of the OLE event's arguments.
ID	Retrieves a long expression that specifies the identifier of the event.
Name	Retrieves the original name of the fired event.
Param	Retrieves an OleEventParam object given either the index of the parameter, or its name.
ToString	Retrieves information about the event.

property OleEvent.CountParam as Long

Retrieves the count of the OLE event's arguments.

Type	Description
Long	A long value that indicates the count of the arguments.

The following sample shows how to enumerate the arguments of an OLE event:

```
Private Sub ExplorerTree1_ItemOleEvent(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal Ev As EXPLOREERTREELibCtl.IOleEvent)
    On Error Resume Next
    Debug.Print "Event name:" & Ev.Name
    If (Ev.CountParam = 0) Then
        Debug.Print "The event has no arguments."
    Else
        Debug.Print "The event has the following arguments:"
        Dim i As Long
        For i = 0 To Ev.CountParam - 1
            Debug.Print Ev(i).Name; " = " & Ev(i).Value
        Next
    End If
End Sub
```

property OleEvent.ID as Long

Retrieves a long expression that specifies the identifier of the event.

Type	Description
Long	A Long expression that defines the identifier of the OLE event.

The identifier of the event could be used to identify a specified OLE event. Use the [Name](#) property of the OLE Event to get the name of the OLE Event. Use the [ToString](#) property to display information about an OLE event. The ToString property displays the identifier of the event after the name of the event in two [] brackets. For instance, the ToString property gets the "KeyDown[-602](KeyCode/Short* = 9,Shift/Short = 0)" when TAB key is pressed, so the identifier of the KeyDown event being fired by the inside User editor is -602.

property OleEvent.Name as String

Retrieves the original name of the fired event.

Type	Description
String	A string expression that indicates the event's name.

Use the Name property to get the name of the event. Use the [ID](#) property to specify a specified even by its identifier. Use the [ToString](#) property to display information about fired event such us name, parameters, types and values. Using the ToString property you can quickly identifies the event that you should handle in your application. The following sample shows how to enumerate the arguments of an OLE event:

```
Private Sub ExplorerTree1_ItemOleEvent(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal Ev As EXPLOREERTREELibCtl.IOleEvent)
    On Error Resume Next
    Debug.Print "Event name:" & Ev.Name
    If (Ev.CountParam = 0) Then
        Debug.Print "The event has no arguments."
    Else
        Debug.Print "The event has the following arguments:"
        Dim i As Long
        For i = 0 To Ev.CountParam - 1
            Debug.Print Ev(i).Name; " = " & Ev(i).Value
        Next
    End If
End Sub
```

property OleEvent.Param (Item as Variant) as OleEventParam

Retrieves an OleEventParam object given either the index of the parameter, or its name.

Type	Description
Item as Variant	A long expression that indicates the argument's index or a string expression that indicates the argument's name.
OleEventParam	An OleEventParam object that contains the name and the value for the argument.

The following sample shows how to enumerate the arguments of an OLE event:

```
Private Sub ExplorerTree1_ItemOleEvent(ByVal Group As EXPLORERTREELibCtl.IGroup,
ByVal Item As EXPLORERTREELibCtl.HITEM, ByVal Ev As EXPLORERTREELibCtl.IOleEvent)
    On Error Resume Next
    Debug.Print "Event name:" & Ev.Name
    If (Ev.CountParam = 0) Then
        Debug.Print "The event has no arguments."
    Else
        Debug.Print "The event has the following arguments:"
        Dim i As Long
        For i = 0 To Ev.CountParam - 1
            Debug.Print Ev(i).Name; " = " & Ev(i).Value
        Next
    End If
End Sub
```

property OleEvent.ToString as String

Retrieves information about the event.

Type	Description
String	A String expression that shows information about an OLE event. The ToString property gets the information as follows: Name[ID] (Param/Type = Value, Param/Type = Value, ...). For instance, "KeyDown[-602] (KeyCode/Short* = 9,Shift/Short = 0)" indicates that the KeyDown event is fired, with the identifier -602 with two parameters KeyCode as a reference to a short type with the value 8, and Shift parameter as Short type with the value 0.

Use the ToString property to display information about fired event such us name, parameters, types and values. Using the ToString property you can quickly identifies the event that you should handle in your application. Use the [ID](#) property to specify a specified even by its identifier. Use the [Name](#) property to get the name of the event. Use the [Param](#) property to access a specified parameter using its index or its name.

Displaying ToString property during the OLE Event event may show data like follows:

```
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseDown[-605](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
KeyDown[-602](KeyCode/Short* = 83,Shift/Short = 0)
KeyPress[-603](KeyAscii/Short* = 115)
Change[2]()
KeyUp[-604](KeyCode/Short* = 83,Shift/Short = 0)
MouseUp[-607](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
```

OleEventParam object

The OleEventParam holds the name and the value for an event's argument.

Name	Description
Name	Retrieves the name of the event's parameter.
Value	Retrieves or sets the value of the event's parameter.

property OleEventParam.Name as String

Retrieves the name of the event's parameter.

Type	Description
String	A string expression that indicates the name of the event's parameter.

The following sample shows how to enumerate the arguments of an OLE event:

```
Private Sub ExplorerTree1_ItemOleEvent(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal Ev As EXPLOREERTREELibCtl.IOleEvent)
    On Error Resume Next
    Debug.Print "Event name:" & Ev.Name
    If (Ev.CountParam = 0) Then
        Debug.Print "The event has no arguments."
    Else
        Debug.Print "The event has the following arguments:"
        Dim i As Long
        For i = 0 To Ev.CountParam - 1
            Debug.Print Ev(i).Name; " = " & Ev(i).Value
        Next
    End If
End Sub
```


property OleEventParam.Value as Variant

Retrieves or sets the value of the event's parameter.

Type	Description
Variant	A variant value that indicates the value of the event's parameter.

The following sample shows how to enumerate the arguments of an OLE event:

```
Private Sub ExplorerTree1_ItemOleEvent(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal Ev As EXPLOREERTREELibCtl.IOleEvent)
    On Error Resume Next
    Debug.Print "Event name:" & Ev.Name
    If (Ev.CountParam = 0) Then
        Debug.Print "The event has no arguments."
    Else
        Debug.Print "The event has the following arguments:"
        Dim i As Long
        For i = 0 To Ev.CountParam - 1
            Debug.Print Ev(i).Name; " = " & Ev(i).Value
        Next
    End If
End Sub
```

ExplorerTree events

The ExplorerTree component supports the following events:

Name	Description
AddColumn	Fired after a new column is added.
AddGroup	Occurs when a new group is added to collection.
AddItem	Occurs after a new Item is inserted to the Items collection.
AfterCellEdit	Occurs after data in the current cell is edited.
AfterExpandGroup	Occurs when a group is expanded or collapsed.
AfterExpandItem	Fired after an item is expanded (collapsed).
AnchorClick	Occurs when an anchor element is clicked.
BeforeCellEdit	Occurs just before the user enters edit mode by clicking in a cell.
BeforeExpandGroup	Occurs just before expanding or collapsing a group.
BeforeExpandItem	Fired before an item is about to be expanded (collapsed).
CellButtonClick	Fired after the user clicks on the cell of button type.
CellImageClick	Fired after the user clicks on the image's cell area.
CellStateChanged	Fired after cell's state is changed.
Click	Occurs when the user presses and then releases the left mouse button over the control.
ColumnClick	Fired after the user clicks on column's header.
DbClick	Occurs when the user dblclk the left mouse button over an object.
ExpandShortcut	Notifies your application that the user just expanded the shortcut bar.
FilterChange	Occurs when filter was changed.
FilterChanging	Occurs just before applying the filter.
FormatColumn	Fired when a cell requires to format its caption.
HyperLinkClick	Occurs when the user clicks on a hyperlink cell.
ItemOleEvent	Fired when an ActiveX control hosted by an item has fired 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.
LayoutChanged	Occurs when control's layout is changed.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
OffsetChanged	Occurs when the scroll position is changed.
OLECompleteDrag	Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled
OLEDragDrop	Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.
OLEDragOver	Occurs when one component is dragged over another.
OLEGiveFeedback	Allows the drag source to specify the type of OLE drag-and-drop operation and the visual feedback.
OLESetData	Occurs on a drag source when a drop target calls the GetData method and there is no data in a specified format in the OLE drag-and-drop DataObject.
OLEStartDrag	Occurs when the OLEDrag method is called.
OversizeChanged	Occurs when the right range of the scroll is changed.
RClick	Fired when right mouse button is clicked
RemoveColumn	Fired before deleting a Column.
RemoveGroup	Fired when a group was removed.
RemoveItem	Occurs before deleting an Item.
ScrollBarClick	Occurs when the user clicks a button in the scrollbar.
SelectGroup	Occurs when a group is clicked.
SelectionChanged	Fired after a new item is selected.
SelectShortcut	Fired when the user selects a new shortcut.
ToolTip	Fired when the control prepares the object's tooltip.

event AddColumn (Group as Group, Column as Column)

Fired after a new column is added.

Type	Description
Group as Group	A Group object where a new column is inserted.
Column as Column	A Column object that's inserted to the group's Columns collection.

The AddColumn event is fired when a new column is inserted to the group's Columns collection. Use the AddColumn event to associate extra data to a new column. Use the [Add](#) method to add new columns to Columns collection. Use the [AddItem](#), [InsertItem](#), [InsertControlItem](#), [PutItems](#) or [DataSource](#) methods to add new items to the group.

Syntax for AddColumn event, **/NET** version, on:

```
C# private void AddColumn(object sender,exontrol.EXPLORERTREELib.Group
Group,exontrol.EXPLORERTREELib.Column Column)
{
}
```

```
VB Private Sub AddColumn(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Column As
exontrol.EXPLORERTREELib.Column) Handles AddColumn
End Sub
```

Syntax for AddColumn event, **/COM** version, on:

```
C# private void AddColumn(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_AddColumnEvent e)
{
}
```

```
C++ void OnAddColumn(LPDISPATCH Group,LPDISPATCH Column)
{
}
```

```
C++ Builder void __fastcall AddColumn(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::IColumn *Column)
{
```

```
}
```

```
Delphi procedure AddColumn(ASender: TObject; Group : IGroup;Column : IColumn);  
begin  
end;
```

```
Delphi 8  
(.NET  
only) procedure AddColumn(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_AddColumnEvent);  
begin  
end;
```

```
Powe... begin event AddColumn(oleobject Group,oleobject Column)  
end event AddColumn
```

```
VB.NET Private Sub AddColumn(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_AddColumnEvent) Handles AddColumn  
End Sub
```

```
VB6 Private Sub AddColumn(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Column As EXPLORERTREELibCtl.IColumn)  
End Sub
```

```
VBA Private Sub AddColumn(ByVal Group As Object,ByVal Column As Object)  
End Sub
```

```
VFP LPARAMETERS Group,Column
```

```
Xbas... PROCEDURE OnAddColumn(oExplorerTree,Group,Column)  
RETURN
```

Syntax for AddColumn event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="AddColumn(Group,Column)" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBSc... <SCRIPT LANGUAGE="VBScript">  
Function AddColumn(Group,Column)
```

```
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComAddColumn Variant IIGroup Variant IIColumn
    Forward Send OnComAddColumn IIGroup IIColumn
End_Procedure
```

```
Visual Objects METHOD OCX_AddColumn(Group,Column) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_AddColumn(COM _Group,COM _Column)
{
}
```

```
XBasic function AddColumn as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Column
as OLE::Exontrol.ExplorerTree.1::IColumn)
end function
```

```
dBASE function nativeObject_AddColumn(Group,Column)
return
```

The following VB sample changes the column's width:

```
Private Sub ExplorerTree1_AddColumn(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal
Column As EXPLORERTREELibCtl.IColumn)
    Column.Width = 196
End Sub
```

event AddGroup (Group as Group)

Occurs when a new group is added to the Groups collection.

Type	Description
Group as Group	A Group object being added.

Use the AddGroup event to notify your application that a new group is to [Groups](#) collection. The [Add](#) method adds a new group to Groups collection. Use the AddGroup event to associate an extra data to the group. Use the [AddItem](#), [InsertItem](#), [InsertControlItem](#), [PutItems](#) or [DataSource](#) methods to add new items to the group.

Syntax for AddGroup event, **/NET** version, on:

```
C# private void AddGroup(object sender,exontrol.EXPLORERTREELib.Group Group)
{
}
```

```
VB Private Sub AddGroup(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group) Handles AddGroup
End Sub
```

Syntax for AddGroup event, **/COM** version, on:

```
C# private void AddGroup(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_AddGroupEvent e)
{
}
```

```
C++ void OnAddGroup(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall AddGroup(TObject *Sender,Explorertreelib_tlb::IGroup *Group)
{
}
```

```
Delphi procedure AddGroup(ASender: TObject; Group : IGroup);
begin
end;
```

```
Delphi 8  
(.NET  
only) procedure AddGroup(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_AddGroupEvent);  
begin  
end;
```

```
Powe... begin event AddGroup(oleobject Group)  
end event AddGroup
```

```
VB.NET Private Sub AddGroup(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_AddGroupEvent) Handles AddGroup  
End Sub
```

```
VB6 Private Sub AddGroup(ByVal Group As EXPLORERTREELibCtl.IGroup)  
End Sub
```

```
VBA Private Sub AddGroup(ByVal Group As Object)  
End Sub
```

```
VFP LPARAMETERS Group
```

```
Xbas... PROCEDURE OnAddGroup(oExplorerTree,Group)  
RETURN
```

Syntax for AddGroup event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="AddGroup(Group)" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">  
Function AddGroup(Group)  
End Function  
</SCRIPT>
```

```
Visual  
Data... Procedure OnComAddGroup Variant IIGroup  
Forward Send OnComAddGroup IIGroup  
End_Procedure
```



```
METHOD OCX_AddGroup(Group) CLASS MainDialog  
RETURN NIL
```

```
X++  
void onEvent_AddGroup(COM _Group)  
{  
}
```

```
XBasic  
function AddGroup as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup)  
end function
```

```
dBASE  
function nativeObject_AddGroup(Group)  
return
```

The following sample changes the background and foreground colors of each group being added:

```
Private Sub ExplorerTree1_AddGroup(ByVal Group As EXPLORERTREELibCtl.IGroup)  
    With Group  
        .BackColor = vbBlue  
        .ForeColor = vbWhite  
    End With  
End Sub
```

event AddItem (Group as Group, Item as HITEM)

Occurs after a new Item is inserted to the Items collection.

Type	Description
Group as Group	A Group object where the newly item is inserted
Item as HITEM	A long expression that indicates the handle of the item that's inserted to the Items collection.

The AddItem event notifies your application that a new items is inserted to the group. Use the [AddItem](#), [InsertItem](#), [InsertControlItem](#), [PutItems](#) or [DataSource](#) methods to add new items to the group. Use the [Add](#) method to add new columns to Columns Collection.

Syntax for AddItem event, **/NET** version, on:

```
C# private void AddItem(object sender,exontrol.EXPLORERTREELib.Group Group,int
Item)
{
}
```

```
VB Private Sub AddItem(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer) Handles AddItem
End Sub
```

Syntax for AddItem event, **/COM** version, on:

```
C# private void AddItem(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_AddItemEvent e)
{
}
```

```
C++ void OnAddItem(LPDISPATCH Group,long Item)
{
}
```

```
C++ Builder void __fastcall AddItem(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::HITEM Item)
{
}
```

Delphi

```
procedure AddItem(ASender: TObject; Group : IGroup;Item : HITEM);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure AddItem(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_AddItemEvent);  
begin  
end;
```

Power...

```
begin event AddItem(oleobject Group,long Item)  
end event AddItem
```

VB.NET

```
Private Sub AddItem(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_AddItemEvent) Handles AddItem  
End Sub
```

VB6

```
Private Sub AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal Item As  
EXPLORERTREELibCtl.HITEM)  
End Sub
```

VBA

```
Private Sub AddItem(ByVal Group As Object,ByVal Item As Long)  
End Sub
```

VFP

```
LPARAMETERS Group,Item
```

Xbas...

```
PROCEDURE OnAddItem(oExplorerTree,Group,Item)  
RETURN
```

Syntax for AddItem event, **/COM** version (others), on:

Java...

```
<SCRIPT EVENT="AddItem(Group,Item)" LANGUAGE="JScript">  
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function AddItem(Group,Item)  
End Function
```

```
</SCRIPT>
```

Visual
Data...

```
Procedure OnComAddItem Variant IIGroup HITEM IItem  
    Forward Send OnComAddItem IIGroup IItem  
End_Procedure
```

Visual
Objects

```
METHOD OCX_AddItem(Group,Item) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_AddItem(COM_Group,int_Item)  
{  
}
```

XBasic

```
function AddItem as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item as  
OLE::Exontrol.ExplorerTree.1::HITEM)  
end function
```

dBASE

```
function nativeObject_AddItem(Group,Item)  
return
```

The following sample changes the bolds the first column in the group, only if the group contains multiple columns:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
    If (Group.Columns.Count > 1) Then  
        With Group.Items  
            .CellBold(Item, 0) = True  
        End With  
    End If  
End Sub
```

event AfterCellEdit (Group as Group, Item as HITEM, ColIndex as Long, NewCaption as String)

Occurs after data in the current cell is edited.

Type	Description
Group as Group	A Group object where the user edits an item.
Item as HITEM	A long expression that indicates the handle of the item being changed.
ColIndex as Long	A long expression that specifies the index of the column where the change occurs, or a handle to a cell being edited if the Item parameter is 0.
NewCaption as String	A string expression that indicates the newly cell's caption.

The AfterCellEdit and [BeforeCellEdit](#) events are fired only if the [AllowEdit](#) property of the group is True. Use the [Edit](#) method to programmatically edits a cell. If the user doesn't handle the AfterCellEdit event the cell's caption remains unchanged. Use the AfterCellEdit event to change the cell's caption after user edits a cell. The AfterCellEdit event is not fired if the user has canceled the edit operation using BeforeCellEdit event.

Syntax for AfterCellEdit event, **/NET** version, on:

```
C# private void AfterCellEdit(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,int ColIndex,string NewCaption)
{
}
```

```
VB Private Sub AfterCellEdit(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal ColIndex As
Integer,ByVal NewCaption As String) Handles AfterCellEdit
End Sub
```

Syntax for AfterCellEdit event, **/COM** version, on:

```
C# private void AfterCellEdit(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_AfterCellEditEvent e)
{
}
```

```
C++ void OnAfterCellEdit(LPDISPATCH Group,long Item,long ColIndex,LPCTSTR
```

```
NewCaption)
```

```
{  
}
```

**C++
Builder**

```
void __fastcall AfterCellEdit(TObject *Sender, ExplorerTreeLib_tlb::IGroup  
*Group, ExplorerTreeLib_tlb::HITEM Item, long ColIndex, BSTR NewCaption)  
{  
}
```

Delphi

```
procedure AfterCellEdit(ASender: TObject; Group : IGroup; Item : HITEM; ColIndex :  
Integer; NewCaption : WideString);  
begin  
end;
```

**Delphi 8
(.NET
only)**

```
procedure AfterCellEdit(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_AfterCellEditEvent);  
begin  
end;
```

Powe...

```
begin event AfterCellEdit(oleobject Group, long Item, long ColIndex, string  
NewCaption)  
end event AfterCellEdit
```

VB.NET

```
Private Sub AfterCellEdit(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_AfterCellEditEvent) Handles  
AfterCellEdit  
End Sub
```

VB6

```
Private Sub AfterCellEdit(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal Item  
As EXPLORERTREELibCtl.HITEM, ByVal ColIndex As Long, ByVal NewCaption As  
String)  
End Sub
```

VBA

```
Private Sub AfterCellEdit(ByVal Group As Object, ByVal Item As Long, ByVal  
ColIndex As Long, ByVal NewCaption As String)  
End Sub
```

VFP

```
LPARAMETERS Group, Item, ColIndex, NewCaption
```

```
Xbas... PROCEDURE OnAfterCellEdit(oExplorerTree,Group,Item,ColIndex,NewCaption)
RETURN
```

Syntax for AfterCellEdit event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="AfterCellEdit(Group,Item,ColIndex,NewCaption)"
LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function AfterCellEdit(Group,Item,ColIndex,NewCaption)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComAfterCellEdit Variant IIGroup HITEM IItem Integer IIColIndex
String IINewCaption
Forward Send OnComAfterCellEdit IIGroup IItem IIColIndex IINewCaption
End_Procedure
```

```
Visual Objects METHOD OCX_AfterCellEdit(Group,Item,ColIndex,NewCaption) CLASS
MainDialog
RETURN NIL
```

```
X++ void onEvent_AfterCellEdit(COM _Group,int _Item,int _ColIndex,str _NewCaption)
{
}
```

```
XBasic function AfterCellEdit as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item as
OLE::Exontrol.ExplorerTree.1::HITEM,ColIndex as N,NewCaption as C)
end function
```

```
dBASE function nativeObject_AfterCellEdit(Group,Item,ColIndex,NewCaption)
return
```

The following sample shows how to change the cell's caption when the edit operation ends.

```
Private Sub ExplorerTree1_AfterCellEdit(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long, ByVal NewCaption As
String)
    With Group.Items
        .CellCaption(Item, ColIndex) = NewCaption
    End With
End Sub
```

Use the BeforeCellEdit is you need to cancel editing cells. The following sample shows how to cancel editing of any cell owned by the first column:

```
Private Sub ExplorerTree1_BeforeCellEdit(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long, Value As Variant,
Cancel As Variant)
    Cancel = ColIndex = 0
End Sub
```


event AfterExpandGroup (Group as Group)

Occurs when a group is expanded or collapsed.

Type	Description
Group as Group	A Group object being expanded or collapsed.

Use the AfterExpandGroup event to notify your application that a group is expanded or collapsed. The [BeforeExpandGroup](#) event is fired just before expanding or collapsing a group. Use the [AllowExpand](#) property to disable expanding or collapsing groups when user clicks the group's caption. Use the [DelayScroll](#) property to specifies the delay used for animation during expanding or collapsing. Use the [Expanded](#) property to expand or collapse programmatically a group.

Syntax for AfterExpandGroup event, **/NET** version, on:

```
C# private void AfterExpandGroup(object sender,exontrol.EXPLORERTREELib.Group
Group)
{
}
```

```
VB Private Sub AfterExpandGroup(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group) Handles AfterExpandGroup
End Sub
```

Syntax for AfterExpandGroup event, **/COM** version, on:

```
C# private void AfterExpandGroup(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_AfterExpandGroupEvent e)
{
}
```

```
C++ void OnAfterExpandGroup(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall AfterExpandGroup(TObject *Sender,Explorertreelib_tlb::IGroup
*Group)
{
}
```

```
Delphi procedure AfterExpandGroup(ASender: TObject; Group : IGroup);  
begin  
end;
```

```
Delphi 8 (.NET only) procedure AfterExpandGroup(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_AfterExpandGroupEvent);  
begin  
end;
```

```
Powe... begin event AfterExpandGroup(oleobject Group)  
end event AfterExpandGroup
```

```
VB.NET Private Sub AfterExpandGroup(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_AfterExpandGroupEvent) Handles  
AfterExpandGroup  
End Sub
```

```
VB6 Private Sub AfterExpandGroup(ByVal Group As EXPLORERTREELibCtl.IGroup)  
End Sub
```

```
VBA Private Sub AfterExpandGroup(ByVal Group As Object)  
End Sub
```

```
VFP LPARAMETERS Group
```

```
Xbas... PROCEDURE OnAfterExpandGroup(oExplorerTree,Group)  
RETURN
```

Syntax for AfterExpandGroup event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="AfterExpandGroup(Group)" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBSc... <SCRIPT LANGUAGE="VBScript">  
Function AfterExpandGroup(Group)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComAfterExpandGroup Variant IIGroup  
    Forward Send OnComAfterExpandGroup IIGroup  
End_Procedure
```

Visual
Objects

```
METHOD OCX_AfterExpandGroup(Group) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_AfterExpandGroup(COM_Group)  
{  
}
```

XBasic

```
function AfterExpandGroup as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup)  
end function
```

dBASE

```
function nativeObject_AfterExpandGroup(Group)  
return
```

The following sample displays the caption of group being expanded or collapsed:

```
Private Sub ExplorerTree1_AfterExpandGroup(ByVal Group As  
EXPLORERTREELibCtl.IGroup)  
    Debug.Print Group.Caption  
End Sub
```

event AfterExpandItem (Group as Group, Item as HITEM)

Fired after an item is expanded (collapsed).

Type	Description
Group as Group	A Group object where the item is expanded or collapsed.
Item as HITEM	A long expression that indicates the item's handle that indicates the item expanded or collapsed.

The AfterExpandItem event notifies your application that an item is collapsed or expanded. Use the [ExpandItem](#) method to programmatically expand or collapse an item. Use the [BeforeExpandItem](#) event to cancel expanding or collapsing items.

Syntax for AfterExpandItem event, **/NET** version, on:

```
C# private void AfterExpandItem(object sender, exontrol.EXPLORERTREELib.Group
Group, int Item)
{
}
```

```
VB Private Sub AfterExpandItem(ByVal sender As System.Object, ByVal Group As
exontrol.EXPLORERTREELib.Group, ByVal Item As Integer) Handles
AfterExpandItem
End Sub
```

Syntax for AfterExpandItem event, **/COM** version, on:

```
C# private void AfterExpandItem(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_AfterExpandItemEvent e)
{
}
```

```
C++ void OnAfterExpandItem(LPDISPATCH Group, long Item)
{
}
```

```
C++ Builder void __fastcall AfterExpandItem(TObject *Sender, ExplorerTreeLib_tlb::IGroup
*Group, ExplorerTreeLib_tlb::HITEM Item)
{
}
```

```
Delphi procedure AfterExpandItem(ASender: TObject; Group : IGroup;Item : HITEM);  
begin  
end;
```

```
Delphi 8 (.NET only) procedure AfterExpandItem(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_AfterExpandItemEvent);  
begin  
end;
```

```
Powe... begin event AfterExpandItem(oleobject Group,long Item)  
end event AfterExpandItem
```

```
VB.NET Private Sub AfterExpandItem(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_AfterExpandItemEvent) Handles  
AfterExpandItem  
End Sub
```

```
VB6 Private Sub AfterExpandItem(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
End Sub
```

```
VBA Private Sub AfterExpandItem(ByVal Group As Object,ByVal Item As Long)  
End Sub
```

```
VFP LPARAMETERS Group,Item
```

```
Xbas... PROCEDURE OnAfterExpandItem(oExplorerTree,Group,Item)  
RETURN
```

Syntax for AfterExpandItem event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="AfterExpandItem(Group,Item)" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBSc... <SCRIPT LANGUAGE="VBScript">  
Function AfterExpandItem(Group,Item)  
End Function
```

```
</SCRIPT>
```

```
Visual  
Data... Procedure OnComAfterExpandItem Variant IIGroup HITEM IItem  
Forward Send OnComAfterExpandItem IIGroup IItem  
End_Procedure
```

```
Visual  
Objects METHOD OCX_AfterExpandItem(Group,Item) CLASS MainDialog  
RETURN NIL
```

```
X++ void onEvent_AfterExpandItem(COM_Group,int_Item)  
{  
}
```

```
XBasic function AfterExpandItem as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item  
as OLE::Exontrol.ExplorerTree.1::HITEM)  
end function
```

```
dBASE function nativeObject_AfterExpandItem(Group,Item)  
return
```

The following sample shows how to cancel expanding or collapsing items:

```
Private Sub ExplorerTree1_BeforeExpandItem(ByVal Group As EXPLORERTREELibCtl.IGroup,  
ByVal Item As EXPLORERTREELibCtl.HITEM, Cancel As Variant)  
Cancel = True  
End Sub
```

The following sample prints the item's state when it is expanded or collapsed:

```
Private Sub ExplorerTree1_AfterExpandItem(ByVal Group As EXPLORERTREELibCtl.IGroup,  
ByVal Item As EXPLORERTREELibCtl.HITEM)  
Debug.Print "The " & Item & " item is " & If(Group.Items.ExpandItem(Item), "expanded",  
"collapsed")  
End Sub
```

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 `<a1;youreextradata>anchor`, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "youreextradata". Use the [AnchorFromPoint](#) property to retrieve the identifier of the anchor element from the cursor.

Syntax for AnchorClick event, **/NET** version, on:

```
C# private void AnchorClick(object sender,string AnchorID,string Options)
{
}
```

```
VB Private Sub AnchorClick(ByVal sender As System.Object,ByVal AnchorID As
String,ByVal Options As String) Handles AnchorClick
End Sub
```

Syntax for AnchorClick event, **/COM** version, on:

```
C# private void AnchorClick(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_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:
AxEXPLORERTREELib._IExplorerTreeEvents_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
AxEXPLORERTREELib._IExplorerTreeEvents_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(oExplorerTree,AnchorID,Options)
RETURN
```


Syntax for AnchorClick event, **/COM** version (others), on:

```
Java... <SCRIPT EVENT="AnchorClick(AnchorID,Options)" LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function AnchorClick(AnchorID,Options)
End Function
</SCRIPT>
```

```
Visual
Data... Procedure OnComAnchorClick String IIAnchorID String IIOptions
Forward Send OnComAnchorClick IIAnchorID IIOptions
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 BeforeCellEdit (Group as Group, Item as HITEM, ColIndex as Long, ByRef Value as Variant, ByRef Cancel as Variant)

Occurs just before the user enters edit mode by clicking in a cell.

Type	Description
Group as Group	A Group object where the change occurs.
Item as HITEM	A long expression that indicates the handle of the item being changed.
ColIndex as Long	A long expression that specifies the index of the column where the change occurs, or the handle of the cell being edited if the Item parameter is 0.
Value as Variant	(By Reference) A Variant expression that indicates the edit's caption. By default, the caption of the edit control is the cell's caption. The user can change the text that the edit control displays.
Cancel as Variant	(By Reference) A boolean expression that indicates whether the control cancels the default operation.

The BeforeCellEdit event notifies your application that the user starts editing a cell. Use the [Edit](#) method to programmatically edit a cell. Use the [AllowEdit](#) property to enable edit feature in the group. Use the BeforeCellEdit event to cancel editing cells or to change the edit's caption before it is displayed. Use the [AfterCellEdit](#) to change the cell's caption when the edit operation ends.

Syntax for BeforeCellEdit event, **/NET** version, on:

```
C# private void BeforeCellEdit(object sender, exontrol.EXPLORERTREELib.Group
Group, int Item, int ColIndex, ref object Value, ref object Cancel)
{
}
```

```
VB Private Sub BeforeCellEdit(ByVal sender As System.Object, ByVal Group As
exontrol.EXPLORERTREELib.Group, ByVal Item As Integer, ByVal ColIndex As
Integer, ByRef Value As Object, ByRef Cancel As Object) Handles BeforeCellEdit
End Sub
```

Syntax for BeforeCellEdit event, **/COM** version, on:

```
C# private void BeforeCellEdit(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeCellEditEvent e)
```

```
{  
}
```

C++

```
void OnBeforeCellEdit(LPDISPATCH Group,long Item,long ColIndex,VARIANT FAR*  
Value,VARIANT FAR* Cancel)  
{  
}
```

C++**Builder**

```
void __fastcall BeforeCellEdit(TObject *Sender,Explorertreelib_tlb::IGroup  
*Group,Explorertreelib_tlb::HITEM Item,long ColIndex,Variant * Value,Variant *  
Cancel)  
{  
}
```

Delphi

```
procedure BeforeCellEdit(ASender: TObject; Group : IGroup;Item : HITEM;ColIndex  
: Integer;var Value : OleVariant;var Cancel : OleVariant);  
begin  
end;
```

**Delphi 8
(.NET
only)**

```
procedure BeforeCellEdit(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeCellEditEvent);  
begin  
end;
```

Powe...

```
begin event BeforeCellEdit(oleobject Group,long Item,long ColIndex,any Value,any  
Cancel)  
end event BeforeCellEdit
```

VB.NET

```
Private Sub BeforeCellEdit(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeCellEditEvent) Handles  
BeforeCellEdit  
End Sub
```

VB6

```
Private Sub BeforeCellEdit(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal Item  
As EXPLORERTREELibCtl.HITEM,ByVal ColIndex As Long,Value As Variant,Cancel As  
Variant)  
End Sub
```

VBA

```
Private Sub BeforeCellEdit(ByVal Group As Object,ByVal Item As Long,ByVal ColIndex As Long,Value As Variant,Cancel As Variant)
End Sub
```

VFP

```
LPARAMETERS Group,Item,ColIndex,Value,Cancel
```

Xbas...

```
PROCEDURE OnBeforeCellEdit(oExplorerTree,Group,Item,ColIndex,Value,Cancel)
RETURN
```

Syntax for BeforeCellEdit event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="BeforeCellEdit(Group,Item,ColIndex,Value,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">
Function BeforeCellEdit(Group,Item,ColIndex,Value,Cancel)
End Function
</SCRIPT>
```

Visual
Data...

```
Procedure OnComBeforeCellEdit Variant IIGroup HITEM IItem Integer IColIndex
Variant IValue Variant ICancel
    Forward Send OnComBeforeCellEdit IIGroup IItem IColIndex IValue ICancel
End_Procedure
```

Visual
Objects

```
METHOD OCX_BeforeCellEdit(Group,Item,ColIndex,Value,Cancel) CLASS
MainDialog
RETURN NIL
```

X++

```
void onEvent_BeforeCellEdit(COM _Group,int _Item,int _ColIndex,COMVariant
/*variant*/ _Value,COMVariant /*variant*/ _Cancel)
{
}
```

XBasic

```
function BeforeCellEdit as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item as
```

```
OLE::Exontrol.ExplorerTree.1::HITEM,ColIndex as N,Value as A,Cancel as A)
end function
```

```
dBASE function nativeObject_BeforeCellEdit(Group,Item,ColIndex,Value,Cancel)
return
```

The following sample cancels editing cells on the first column only for the first group:

```
Private Sub ExplorerTree1_BeforeCellEdit(ByVal Group As EXPLORERTREELibCtl.IGroup,
ByVal Item As EXPLORERTREELibCtl.HITEM, ByVal ColIndex As Long, Value As Variant,
Cancel As Variant)
    If (Group.Index = 0) Then
        Cancel = ColIndex = 0
    End If
End Sub
```

event BeforeExpandGroup (Group as Group, ByRef Cancel as Variant)

Occurs just before expanding or collapsing a group.

Type	Description
Group as Group	A Group object being expanded or collapsed.
Cancel as Variant	(By Reference) A boolean expression that indicates whether the expanding/collapsing operation is canceled.

Use the BeforeExpandGroup event to disable expanding groups on the fly. Use the [AllowExpand](#) property to disable expanding or collapsing groups when user clicks the group's caption. The BeforeExpandGroup event notifies your application that a group is expanding or collapsing. The control fires the [AfterExpandGroup](#) event after group is expanded or collapsed. Use the [Expanded](#) property to expand programmatically the group

Syntax for BeforeExpandGroup event, **/NET** version, on:

```
C# private void BeforeExpandGroup(object sender,exontrol.EXPLORERTREELib.Group
Group,ref object Cancel)
{
}
```

```
VB Private Sub BeforeExpandGroup(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByRef Cancel As Object) Handles
BeforeExpandGroup
End Sub
```

Syntax for BeforeExpandGroup event, **/COM** version, on:

```
C# private void BeforeExpandGroup(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeExpandGroupEvent e)
{
}
```

```
C++ void OnBeforeExpandGroup(LPDISPATCH Group,VARIANT FAR* Cancel)
{
}
```

```
C++ Builder void __fastcall BeforeExpandGroup(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Variant * Cancel)
```

```
{  
}
```

Delphi
procedure BeforeExpandGroup(ASender: TObject; Group : IGroup;var Cancel : OleVariant);
begin
end;

**Delphi 8
(.NET
only)**
procedure BeforeExpandGroup(sender: System.Object; e:
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeExpandGroupEvent);
begin
end;

Powe...
begin event BeforeExpandGroup(oleobject Group,any Cancel)
end event BeforeExpandGroup

VB.NET
Private Sub BeforeExpandGroup(ByVal sender As System.Object, ByVal e As
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeExpandGroupEvent) Handles
BeforeExpandGroup
End Sub

VB6
Private Sub BeforeExpandGroup(ByVal Group As
EXPLORERTREELibCtl.IGroup,Cancel As Variant)
End Sub

VBA
Private Sub BeforeExpandGroup(ByVal Group As Object,Cancel As Variant)
End Sub

VFP
LPARAMETERS Group,Cancel

Xbas...
PROCEDURE OnBeforeExpandGroup(oExplorerTree,Group,Cancel)
RETURN

Syntax for BeforeExpandGroup event, **ICOM** version (others), on:

Java...
<SCRIPT EVENT="BeforeExpandGroup(Group,Cancel)" LANGUAGE="JScript">
</SCRIPT>

```
VBS... <SCRIPT LANGUAGE="VBScript">  
Function BeforeExpandGroup(Group,Cancel)  
End Function  
</SCRIPT>
```

```
Visual  
Data... Procedure OnComBeforeExpandGroup Variant IIGroup Variant IICancel  
Forward Send OnComBeforeExpandGroup IIGroup IICancel  
End_Procedure
```

```
Visual  
Objects METHOD OCX_BeforeExpandGroup(Group,Cancel) CLASS MainDialog  
RETURN NIL
```

```
X++ void onEvent_BeforeExpandGroup(COM _Group,COMVariant /*variant*/ _Cancel)  
{  
}
```

```
XBasic function BeforeExpandGroup as v (Group as  
OLE::Exontrol.ExplorerTree.1::IGroup,Cancel as A)  
end function
```

```
dBASE function nativeObject_BeforeExpandGroup(Group,Cancel)  
return
```

The following sample disables expanding or collapsing the second group:

```
Private Sub ExplorerTree1_BeforeExpandGroup(ByVal Group As  
EXPLORERTREELibCtl.IGroup, Cancel As Variant)  
Cancel = Group.Index = 1  
End Sub
```


event BeforeExpandItem (Group as Group, Item as HITEM, ByRef Cancel as Variant)

Fired before an item is about to be expanded (collapsed).

Type	Description
Group as Group	A Group object where the user expands or collapse an item.
Item as HITEM	A long expression that indicates the handle of the item being expanded or collapsed.
Cancel as Variant	(By Reference) A boolean expression that indicates whether the control cancel expanding or collapsing the item.

The BeforeExpandItem event notifies your application that an item is about to be collapsed or expanded. Use the BeforeExpandItem event to cancel expanding or collapsing items. The [AfterExapandItem](#) event is fired after an item is expanded or collapsed. Use the [ExpandItem](#) method to programmatically expand or collapse an item.

Syntax for BeforeExpandItem event, **/NET** version, on:

```
C# private void BeforeExpandItem(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,ref object Cancel)
{
}
```

```
VB Private Sub BeforeExpandItem(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByRef Cancel As Object)
Handles BeforeExpandItem
End Sub
```

Syntax for BeforeExpandItem event, **/COM** version, on:

```
C# private void BeforeExpandItem(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeExpandItemEvent e)
{
}
```

```
C++ void OnBeforeExpandItem(LPDISPATCH Group,long Item,VARIANT FAR* Cancel)
{
```

```
}
```

**C++
Builder**

```
void __fastcall BeforeExpandItem(TObject *Sender,ExplorerTreeLib_tlb::IGroup  
*Group,ExplorerTreeLib_tlb::HITEM Item,Variant * Cancel)  
{  
}
```

Delphi

```
procedure BeforeExpandItem(ASender: TObject; Group : IGroup;Item : HITEM;var  
Cancel : OleVariant);  
begin  
end;
```

**Delphi 8
(.NET
only)**

```
procedure BeforeExpandItem(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeExpandItemEvent);  
begin  
end;
```

Powe...

```
begin event BeforeExpandItem(oleobject Group,long Item,any Cancel)  
end event BeforeExpandItem
```

VB.NET

```
Private Sub BeforeExpandItem(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_BeforeExpandItemEvent) Handles  
BeforeExpandItem  
End Sub
```

VB6

```
Private Sub BeforeExpandItem(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Item As EXPLORERTREELibCtl.HITEM,Cancel As Variant)  
End Sub
```

VBA

```
Private Sub BeforeExpandItem(ByVal Group As Object,ByVal Item As Long,Cancel  
As Variant)  
End Sub
```

VFP

```
LPARAMETERS Group,Item,Cancel
```

Xbas...

```
PROCEDURE OnBeforeExpandItem(oExplorerTree,Group,Item,Cancel)  
RETURN
```

Syntax for BeforeExpandItem event, **/COM** version (others), on:

```
Java... <SCRIPT EVENT="BeforeExpandItem(Group,Item,Cancel)" LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function BeforeExpandItem(Group,Item,Cancel)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComBeforeExpandItem Variant IIGroup HITEM IItem Variant IICancel
Forward Send OnComBeforeExpandItem IIGroup IItem IICancel
End_Procedure
```

```
Visual Objects METHOD OCX_BeforeExpandItem(Group,Item,Cancel) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_BeforeExpandItem(COM _Group,int _Item,COMVariant /*variant*/
_Cancel)
{
}
```

```
XBasic function BeforeExpandItem as v (Group as
OLE::Exontrol.ExplorerTree.1::IGroup,Item as
OLE::Exontrol.ExplorerTree.1::HITEM,Cancel as A)
end function
```

```
dBASE function nativeObject_BeforeExpandItem(Group,Item,Cancel)
return
```

The following sample cancels expanding or collapsing items in the first group:

```
Private Sub ExplorerTree1_BeforeExpandItem(ByVal Group As EXPLORERTREELibCtl.IGroup,
ByVal Item As EXPLORERTREELibCtl.HITEM, Cancel As Variant)
If (Group.Index = 0) Then
Cancel = True
End If
```

If your application expands items programmatically and you need to disable expanding or collapsing items only when user tries to click the left button of each parent item you can use an internal counter like in the following sample:

```
Dim iExpanding As Long
```

```
iExpanding = 0
```

```
iExpanding = iExpanding + 1
```

```
....
```

```
do expanding/collapsing items
```

```
...
```

```
iExpanding = iExpanding - 1
```

```
Private Sub ExplorerTree1_BeforeExpandItem(ByVal Group As EXPLORERTREELibCtl.IGroup,  
ByVal Item As EXPLORERTREELibCtl.HITEM, Cancel As Variant)
```

```
    If iExpanding = 0 Then
```

```
        If (Group.Index = 0) Then
```

```
            Cancel = True
```

```
        End If
```

```
    End If
```

```
End Sub
```

event CellButtonClick (Group as Group, Item as HITEM, ColIndex as Long)

Fired after the user clicks on the cell of button type.

Type	Description
Group as Group	A Group object where user clicks a button.
Item as HITEM	A long expression that indicates the handle of the item where the user clicks the cell's button.
ColIndex as Long	A long expression that specifies the index of the column where the user clicks the cell's button, or a long expression that indicates the handle of the cell being clicked, if the Item parameter is 0.

The CellButtonClick event is fired after the user released the left mouse button over a cell of button type. Use the [CellHasButton](#) property to specify whether a cell is of button type. The CellButtonClick event notifies your application that user presses a cell of the button type.

The following sample prints the caption of the clicked cell:

Syntax for CellButtonClick event, **/NET** version, on:

```
C# private void CellButtonClick(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,int ColIndex)
{
}
```

```
VB Private Sub CellButtonClick(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal ColIndex As Integer)
Handles CellButtonClick
End Sub
```

Syntax for CellButtonClick event, **/COM** version, on:

```
C# private void CellButtonClick(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_CellButtonClickEvent e)
{
}
```

```
C++ void OnCellButtonClick(LPDISPATCH Group,long Item,long ColIndex)
{
```

```
}
```

**C++
Builder**

```
void __fastcall CellButtonClick(TObject *Sender,Explorertreelib_tlb::IGroup  
*Group,Explorertreelib_tlb::HITEM Item,long ColIndex)  
{  
}
```

Delphi

```
procedure CellButtonClick(ASender: TObject; Group : IGroup;Item :  
HITEM;ColIndex : Integer);  
begin  
end;
```

**Delphi 8
(.NET
only)**

```
procedure CellButtonClick(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_CellButtonClickEvent);  
begin  
end;
```

Powe...

```
begin event CellButtonClick(oleobject Group,long Item,long ColIndex)  
end event CellButtonClick
```

VB.NET

```
Private Sub CellButtonClick(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_CellButtonClickEvent) Handles  
CellButtonClick  
End Sub
```

VB6

```
Private Sub CellButtonClick(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Item As EXPLORERTREELibCtl.HITEM,ByVal ColIndex As Long)  
End Sub
```

VBA

```
Private Sub CellButtonClick(ByVal Group As Object,ByVal Item As Long,ByVal  
ColIndex As Long)  
End Sub
```

VFP

```
LPARAMETERS Group,Item,ColIndex
```

Xbas...

```
PROCEDURE OnCellButtonClick(oExplorerTree,Group,Item,ColIndex)  
RETURN
```

Syntax for CellButtonClick event, **/COM** version (others), on:

```
Java... <SCRIPT EVENT="CellButtonClick(Group,Item,ColIndex)" LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function CellButtonClick(Group,Item,ColIndex)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComCellButtonClick Variant IIGroup HITEM IItem Integer IIColIndex
Forward Send OnComCellButtonClick IIGroup IItem IIColIndex
End_Procedure
```

```
Visual Objects METHOD OCX_CellButtonClick(Group,Item,ColIndex) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_CellButtonClick(COM _Group,int _Item,int _ColIndex)
{
}
```

```
XBasic function CellButtonClick as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item
as OLE::Exontrol.ExplorerTree.1::HITEM,ColIndex as N)
end function
```

```
dBASE function nativeObject_CellButtonClick(Group,Item,ColIndex)
return
```

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal
Item As EXPLOREERTREELibCtl.HITEM)
With Group.Items
.CellHasButton(Item, 0) = True
End With
End Sub
```

```
Private Sub ExplorerTree1_CellButtonClick(ByVal Group As EXPLOREERTREELibCtl.IGroup,
```

```
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)
```

```
    MsgBox Group.Items.CellCaption(Item, ColIndex)
```

```
End Sub
```


event CellImageClick (Group as Group, Item as HITEM, ColIndex as Long)

Fired after the user clicks on the image's cell area.

Type	Description
Group as Group	A Group object where the user clicks an icon.
Item as HITEM	A long expression that indicates the handle of the item where the user clicks the cell's icon.
ColIndex as Long	A long expression that indicates the index of the column where the user clicks the cell's icon, or a long expression that indicates the handle of the cell being clicked, if the Item parameter is 0.

The CellImageClick event is fired when user clicks on the cell's icon. Use the [CellImage](#) property to assign an icon to a cell. Use the [CellImages](#) property to assign multiple icons to a cell.

Syntax for CellImageClick event, **/NET** version, on:

```
C# private void CellImageClick(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,int ColIndex)
{
}
```

```
VB Private Sub CellImageClick(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal ColIndex As Integer)
Handles CellImageClick
End Sub
```

Syntax for CellImageClick event, **/COM** version, on:

```
C# private void CellImageClick(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_CellImageClickEvent e)
{
}
```

```
C++ void OnCellImageClick(LPDISPATCH Group,long Item,long ColIndex)
{
}
```

C++
Builder

```
void __fastcall CellImageClick(TObject *Sender,ExplorerTreeLib_tlb::IGroup  
*Group,ExplorerTreeLib_tlb::HITEM Item,long ColIndex)  
{  
}
```

Delphi

```
procedure CellImageClick(ASender: TObject; Group : IGroup;Item : HITEM;ColIndex  
: Integer);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure CellImageClick(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_CellImageClickEvent);  
begin  
end;
```

Powe...

```
begin event CellImageClick(oleobject Group,long Item,long ColIndex)  
end event CellImageClick
```

VB.NET

```
Private Sub CellImageClick(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_CellImageClickEvent) Handles  
CellImageClick  
End Sub
```

VB6

```
Private Sub CellImageClick(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Item As EXPLORERTREELibCtl.HITEM,ByVal ColIndex As Long)  
End Sub
```

VBA

```
Private Sub CellImageClick(ByVal Group As Object,ByVal Item As Long,ByVal  
ColIndex As Long)  
End Sub
```

VFP

```
LPARAMETERS Group,Item,ColIndex
```

Xbas...

```
PROCEDURE OnCellImageClick(oExplorerTree,Group,Item,ColIndex)  
RETURN
```

Syntax for CellImageClick event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="CellImageClick(Group,Item,ColIndex)" LANGUAGE="JScript" >
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript" >
Function CellImageClick(Group,Item,ColIndex)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComCellImageClick Variant IIGroup HITEM IItem Integer IColIndex
Forward Send OnComCellImageClick IIGroup IItem IColIndex
End_Procedure
```

```
Visual Objects METHOD OCX_CellImageClick(Group,Item,ColIndex) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_CellImageClick(COM _Group,int _Item,int _ColIndex)
{
}
```

```
XBasic function CellImageClick as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item as
OLE::Exontrol.ExplorerTree.1::HITEM,ColIndex as N)
end function
```

```
dBASE function nativeObject_CellImageClick(Group,Item,ColIndex)
return
```

The following sample displays the caption of the cell whose icon is clicked:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal
Item As EXPLORERTREELibCtl.HITEM)
With Group.Items
.CellImage(Item, 0) = 1
End With
End Sub
```

```
Private Sub ExplorerTree1_CellImageClick(ByVal Group As EXPLOREERTREELibCtl.IGroup,  
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)  
    With Group.Items  
        MsgBox .CellCaption(Item, ColIndex)  
    End With  
End Sub
```

event CellStateChanged (Group as Group, Item as HITEM, ColIndex as Long)

Fired after cell's state is changed.

Type	Description
Group as Group	A Group object where the user clicks a check box cell.
Item as HITEM	A long expression that indicates the handle of the item where the cell's state is changed.
ColIndex as Long	A long expression that indicates the index of the column where the cell's state is changed, or a long expression that indicates the handle of the cell, if the Item parameter is 0.

A cell that contains a radio button or a check box button fires the CellStateChanged event when its state is changed. Use the [CellState](#) property to change the cell's state. Use the [CellHasRadioButton](#) or [CellHasCheckBox](#) property to enable radio or check box button into a cell.

Syntax for CellStateChanged event, **/NET** version, on:

```
C# private void CellStateChanged(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,int ColIndex)
{
}
```

```
VB Private Sub CellStateChanged(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal ColIndex As Integer)
Handles CellStateChanged
End Sub
```

Syntax for CellStateChanged event, **/COM** version, on:

```
C# private void CellStateChanged(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_CellStateChangedEvent e)
{
}
```

```
C++ void OnCellStateChanged(LPDISPATCH Group,long Item,long ColIndex)
{
```

```
}
```

**C++
Builder**

```
void __fastcall CellStateChanged(TObject *Sender,Explorerlib_tlb::IGroup  
*Group,Explorerlib_tlb::HITEM Item,long ColIndex)  
{  
}
```

Delphi

```
procedure CellStateChanged(ASender: TObject; Group : IGroup;Item :  
HITEM;ColIndex : Integer);  
begin  
end;
```

**Delphi 8
(.NET
only)**

```
procedure CellStateChanged(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_CellStateChangedEvent);  
begin  
end;
```

Powe...

```
begin event CellStateChanged(oleobject Group,long Item,long ColIndex)  
end event CellStateChanged
```

VB.NET

```
Private Sub CellStateChanged(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_CellStateChangedEvent) Handles  
CellStateChanged  
End Sub
```

VB6

```
Private Sub CellStateChanged(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Item As EXPLORERTREELibCtl.HITEM,ByVal ColIndex As Long)  
End Sub
```

VBA

```
Private Sub CellStateChanged(ByVal Group As Object,ByVal Item As Long,ByVal  
ColIndex As Long)  
End Sub
```

VFP

```
LPARAMETERS Group,Item,ColIndex
```

Xbas...

```
PROCEDURE OnCellStateChanged(oExplorerTree,Group,Item,ColIndex)  
RETURN
```

Syntax for CellStateChanged event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="CellStateChanged(Group,Item,ColIndex)"
LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function CellStateChanged(Group,Item,ColIndex)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComCellStateChanged Variant IIGroup HITEM IItem Integer
IIColIndex
Forward Send OnComCellStateChanged IIGroup IItem IIColIndex
End_Procedure
```

```
Visual Objects METHOD OCX_CellStateChanged(Group,Item,ColIndex) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_CellStateChanged(COM _Group,int _Item,int _ColIndex)
{
}
```

```
XBasic function CellStateChanged as v (Group as
OLE::Exontrol.ExplorerTree.1::IGroup,Item as
OLE::Exontrol.ExplorerTree.1::HITEM,ColIndex as N)
end function
```

```
dBASE function nativeObject_CellStateChanged(Group,Item,ColIndex)
return
```

The following sample displays the cell's caption whose check box is clicked:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLOREERTREELibCtl.IGroup, ByVal
Item As EXPLOREERTREELibCtl.HITEM)
With Group.Items
.CellHasCheckBox(Item, 0) = True
```

```
End With  
End Sub
```

```
Private Sub ExplorerTree1_CellStateChanged(ByVal Group As EXPLOREERTREELibCtl.IGroup,  
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long)
```

```
With Group.Items
```

```
    Debug.Print .CellCaption(Item, ColIndex)
```

```
End With
```

```
End Sub
```


event Click ()

Occurs when the user presses and then releases the left mouse button over the control.

Type

Description

The Click event is fired when the user releases the left mouse button over the control. Use a [MouseDown](#) or [MouseUp](#) event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and [DbClick](#) 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 [ExpandOnClick](#) property expands the group when its caption is clicked.

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;
```

Power...
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(oExplorerTree)
RETURN

Syntax for Click event, **ICOM** 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 ColumnClick (Group as Group, Column as Column)

Fired after the user clicks on column's header.

Type	Description
Group as Group	A Group object where the user clicks the column's caption.
Column as Column	A Column object being clicked.

The ColumnClick event is fired when the user clicks the column's header. By default, the control sorts by the column when user clicks the column's header. Use the [SortOnClick](#) property to specify the operation that control does when user clicks the column's caption. Use the [Columns](#) property to access the group's Columns collection.

Syntax for ColumnClick event, **/NET** version, on:

```
C# private void ColumnClick(object sender,exontrol.EXPLORERTREELib.Group
Group,exontrol.EXPLORERTREELib.Column Column)
{
}
```

```
VB Private Sub ColumnClick(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Column As
exontrol.EXPLORERTREELib.Column) Handles ColumnClick
End Sub
```

Syntax for ColumnClick event, **/COM** version, on:

```
C# private void ColumnClick(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_ColumnClickEvent e)
{
}
```

```
C++ void OnColumnClick(LPDISPATCH Group,LPDISPATCH Column)
{
}
```

```
C++ Builder void __fastcall ColumnClick(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::IColumn *Column)
{
}
```

```
Delphi procedure ColumnClick(ASender: TObject; Group : IGroup;Column : IColumn);  
begin  
end;
```

```
Delphi 8 (.NET only) procedure ColumnClick(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_ColumnClickEvent);  
begin  
end;
```

```
Powe... begin event ColumnClick(oleobject Group,oleobject Column)  
end event ColumnClick
```

```
VB.NET Private Sub ColumnClick(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_ColumnClickEvent) Handles  
ColumnClick  
End Sub
```

```
VB6 Private Sub ColumnClick(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Column As EXPLORERTREELibCtl.IColumn)  
End Sub
```

```
VBA Private Sub ColumnClick(ByVal Group As Object,ByVal Column As Object)  
End Sub
```

```
VFP LPARAMETERS Group,Column
```

```
Xbas... PROCEDURE OnColumnClick(oExplorerTree,Group,Column)  
RETURN
```

Syntax for ColumnClick event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="ColumnClick(Group,Column)" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBSc... <SCRIPT LANGUAGE="VBScript">  
Function ColumnClick(Group,Column)  
End Function
```

```
</SCRIPT>
```

Visual
Data...

```
Procedure OnComColumnClick Variant IIGroup Variant IIColumn  
    Forward Send OnComColumnClick IIGroup IIColumn  
End_Procedure
```

Visual
Objects

```
METHOD OCX_ColumnClick(Group,Column) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_ColumnClick(COM _Group,COM _Column)  
{  
}
```

XBasic

```
function ColumnClick as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Column  
as OLE::Exontrol.ExplorerTree.1::IColumn)  
end function
```

dBASE

```
function nativeObject_ColumnClick(Group,Column)  
return
```

The following sample displays the caption of the column being clicked:

```
Private Sub ExplorerTree1_ColumnClick(ByVal Group As EXPLORERTREELibCtl.IGroup,  
ByVal Column As EXPLORERTREELibCtl.IColumn)  
    Debug.Print "The '" & Group.Caption & "' & Column.Caption & "' column is clicked."  
End Sub
```

event DbClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user dblclk the left mouse button over an object.

Type	Description
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The DbClick event is fired when user double clicks the control. Use the [GroupFromPoint](#) property to get the group's caption from the point. Use the [GroupListFromPoint](#) property to get the group's list from point. Use the [ItemFromPoint](#) property to get the item over the cursor. Use the [ColumnFromPoint](#) property to get the column's caption over the cursor. The [ExpandOnDbClick](#) property specifies whether the item is expanded or collapsed if the user dbl clicks the item. The [ExpandOnClick](#) property expands the group when its caption is clicked.

Syntax for DbClick event, **/NET** version, on:

```
C# private void DbClick(object sender,short Shift,int X,int Y)
{
}
```

```
VB Private Sub DbClick(ByVal sender As System.Object,ByVal Shift As Short,ByVal X
As Integer,ByVal Y As Integer) Handles DbClick
End Sub
```

Syntax for DbClick event, **/COM** version, on:

```
C# private void DbClick(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_DbClickEvent e)
{
}
```

```
C++ void OnDbClick(short Shift,long X,long Y)
```

```
{  
}
```

C++
Builder

```
void __fastcall DbClick(TObject *Sender,short Shift,int X,int Y)  
{  
}
```

Delphi

```
procedure DbClick(ASender: TObject; Shift : Smallint;X : Integer;Y : Integer);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure DbClick(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_DblClickEvent);  
begin  
end;
```

Power...

```
begin event DbClick(integer Shift,long X,long Y)  
end event DbClick
```

VB.NET

```
Private Sub DbClick(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_DblClickEvent) Handles DbClick  
End Sub
```

VB6

```
Private Sub DbClick(Shift As Integer,X As Single,Y As Single)  
End Sub
```

VBA

```
Private Sub DbClick(ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)  
End Sub
```

VFP

```
LPARAMETERS Shift,X,Y
```

Xbas...

```
PROCEDURE OnDbClick(oExplorerTree,Shift,X,Y)  
RETURN
```

Syntax for DbClick event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="DbClick(Shift,X,Y)" LANGUAGE="JScript">
```



```
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function DbClick(Shift,X,Y)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComDbClick Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS  
IYY  
    Forward Send OnComDbClick IIShift IIX IYY  
End_Procedure
```

Visual
Objects

```
METHOD OCX_DbClick(Shift,X,Y) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_DbClick(int _Shift,int _X,int _Y)  
{  
}
```

XBasic

```
function DbClick as v (Shift as N,X as  
OLE::Exontrol.ExplorerTree.1::OLE_XPOS_PIXELS,Y as  
OLE::Exontrol.ExplorerTree.1::OLE_YPOS_PIXELS)  
end function
```

dBASE

```
function nativeObject_DbClick(Shift,X,Y)  
return
```

The following sample expands a group only when user double clicks the group's caption (The sample requires ExpandOnClick property on False):

```
Private Sub ExplorerTree1_DbClick(Shift As Integer, X As Single, Y As Single)  
    With ExplorerTree1  
        Dim g As EXPLORERTREELibCtl.Group  
        Set g = .GroupFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)  
        If Not g Is Nothing Then  
            g.Expanded = Not g.Expanded  
        End If  
    End With
```


event ExpandShortcut (OldCount as Long, NewCount as Long)

Notifies your application that the user just expanded the shortcut bar.

Type	Description
OldCount as Long	A long expression that indicates the count of expanded shortcuts before expanding the shortcut bar
NewCount as Long	A long expression that indicates the count of expanded shortcuts after expanding the shortcut bar. The NewCount parameter is the same as ExpandShortcutCount property.

The ExpandShortcut event notifies your application when the user resizes the control's shortcut bar. The [ExpandShortcutCount](#) property retrieves or sets a value that indicates the number of shortcuts being expanded. Use the [ExpandShortcutCount](#) property to programmatically expand the shortcut bar. Use the [ShortcutFromPoint](#) property to retrieve the shortcut from the cursor. Use the [ShowShortcutBar](#) property to show or hide the control's shortcut bar. Use the [AllowResizeShortcutBar](#) property to enable or disable resizing the shortcut bar. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut.

Syntax for ExpandShortcut event, **/NET** version, on:

```
C# private void ExpandShortcut(object sender,int OldCount,int NewCount)
{
}
```

```
VB Private Sub ExpandShortcut(ByVal sender As System.Object,ByVal OldCount As Integer,ByVal NewCount As Integer) Handles ExpandShortcut
End Sub
```

Syntax for ExpandShortcut event, **/COM** version, on:

```
C# private void ExpandShortcut(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_ExpandShortcutEvent e)
{
}
```

```
C++ void OnExpandShortcut(long OldCount,long NewCount)
{
}
```

```
C++ Builder void __fastcall ExpandShortcut(TObject *Sender,long OldCount,long NewCount)
{
}
```

```
Delphi procedure ExpandShortcut(ASender: TObject; OldCount : Integer;NewCount : Integer);
begin
end;
```

```
Delphi 8 (.NET only) procedure ExpandShortcut(sender: System.Object; e: AxEXPLORERTREELib._IExplorerTreeEvents_ExpandShortcutEvent);
begin
end;
```

```
Powe... begin event ExpandShortcut(long OldCount,long NewCount)
end event ExpandShortcut
```

```
VB.NET Private Sub ExpandShortcut(ByVal sender As System.Object, ByVal e As AxEXPLORERTREELib._IExplorerTreeEvents_ExpandShortcutEvent) Handles ExpandShortcut
End Sub
```

```
VB6 Private Sub ExpandShortcut(ByVal OldCount As Long,ByVal NewCount As Long)
End Sub
```

```
VBA Private Sub ExpandShortcut(ByVal OldCount As Long,ByVal NewCount As Long)
End Sub
```

```
VFP LPARAMETERS OldCount,NewCount
```

```
Xbas... PROCEDURE OnExpandShortcut(oExplorerTree,OldCount,NewCount)
RETURN
```

Syntax for ExpandShortcut event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="ExpandShortcut(OldCount,NewCount)" LANGUAGE="JScript">
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function ExpandShortcut(OldCount,NewCount)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComExpandShortcut Integer IIOldCount Integer IINewCount  
Forward Send OnComExpandShortcut IIOldCount IINewCount  
End_Procedure
```

Visual
Objects

```
METHOD OCX_ExpandShortcut(OldCount,NewCount) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_ExpandShortcut(int _OldCount,int _NewCount)  
{  
}
```

XBasic

```
function ExpandShortcut as v (OldCount as N,NewCount as N)  
end function
```

dBASE

```
function nativeObject_ExpandShortcut(OldCount,NewCount)  
return
```

event FilterChange (Group as Group)

Occurs when filter was changed.

Type	Description
Group as Group	A Group object where the filter is changed.

Use the FilterChange event to notify your application that the group's filter is changed. The [ApplyFilter](#) and [ClearFilter](#) methods fire the FilterChange event. Use the [DisplayFilterButton](#) property to display the column's filter button.

Syntax for FilterChange event, **/NET** version, on:

```
C# private void FilterChange(object sender,exontrol.EXPLORERTREELib.Group Group)
{
}
```

```
VB Private Sub FilterChange(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group) Handles FilterChange
End Sub
```

Syntax for FilterChange event, **/COM** version, on:

```
C# private void FilterChange(object sender,
AxEXPLORERTREELib._IexplorerTreeEvents_FilterChangeEvent e)
{
}
```

```
C++ void OnFilterChange(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall FilterChange(TObject *Sender,Explorertreelib_tlb::IGroup *Group)
{
}
```

```
Delphi procedure FilterChange(ASender: TObject; Group : IGroup);
begin
end;
```

Delphi 8
(.NET
only)

```
procedure FilterChange(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_FilterChangeEvent);  
begin  
end;
```

Powe...
begin event FilterChange(oleobject Group)
end event FilterChange

VB.NET
Private Sub FilterChange(ByVal sender As System.Object, ByVal e As
AxEXPLORERTREELib._IExplorerTreeEvents_FilterChangeEvent) Handles
FilterChange
End Sub

VB6
Private Sub FilterChange(ByVal Group As EXPLORERTREELibCtl.IGroup)
End Sub

VBA
Private Sub FilterChange(ByVal Group As Object)
End Sub

VFP
LPARAMETERS Group

Xbas...
PROCEDURE OnFilterChange(oExplorerTree,Group)
RETURN

Syntax for FilterChange event, **ICOM** version (others), on:

Java...
<SCRIPT EVENT="FilterChange(Group)" LANGUAGE="JScript">
</SCRIPT>

VBSc...
<SCRIPT LANGUAGE="VBScript">
Function FilterChange(Group)
End Function
</SCRIPT>

Visual
Data...

```
Procedure OnComFilterChange Variant IIGroup  
    Forward Send OnComFilterChange IIGroup  
End_Procedure
```

Visual
Objects

```
METHOD OCX_FilterChange(Group) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_FilterChange(COM_Group)  
{  
}
```

XBasic

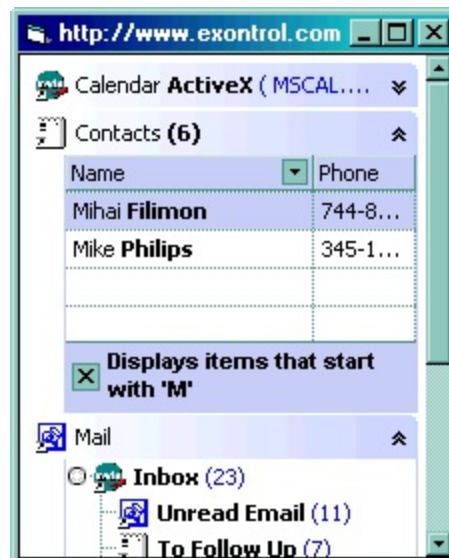
```
function FilterChange as v (Group as OLE::Exontrol.ExplorerTree.1::IIGroup)  
end function
```

dBASE

```
function nativeObject_FilterChange(Group)  
return
```

The following sample displays a new caption on the group's filter bar when the user changes the group's filter:

```
Private Sub ExplorerTree1_FilterChange(ByVal Group As EXPLORERTREELibCtl.IIGroup)  
    Group.FilterBarCaption = "Displays items that start with 'M'"  
End Sub
```



event FilterChanging (Group as Group)

Notifies your application that the filter for a group is about to change

Type	Description
Group as Group	A Group object where the filter is about to be change.

The FilterChanging event occurs just before applying the filter. The [FilterChange](#) event occurs once the filter is applied, so the list gets filtered. Use the [Filter](#) and [FilterType](#) properties to retrieve the column's filter string, if case, and the column's filter type. The [ApplyFilter](#) and [ClearFilter](#) methods fire the FilterChange event. Use the [DisplayFilterButton](#) property to add a filter bar button to the column's caption. Use the [FilterBarHeight](#) property to specify the height of the control's filter bar. Use the [FilterBarFont](#) property to specify the font for the control's filter bar. For instance, you can use the FilterChanging event to start a timer, and count the time to get the filter applied, when the FilterChange event is fired. The [FilterBarPromptVisible](#) property specifies whether the filter prompt is visible or hidden. 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.

Syntax for FilterChanging event, **/NET** version, on:

```
C# private void FilterChanging(object sender,exontrol.EXPLORERTREELib.Group
Group)
{
}
```

```
VB Private Sub FilterChanging(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group) Handles FilterChanging
End Sub
```

Syntax for FilterChanging event, **/COM** version, on:

```
C# private void FilterChanging(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_FilterChangingEvent e)
{
}
```

```
C++ void OnFilterChanging(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall FilterChanging(TObject *Sender, ExplorerTreeLib_tlb::IGroup *Group)
{
}
```

```
Delphi procedure FilterChanging(ASender: TObject; Group : IGroup);
begin
end;
```

```
Delphi 8 (.NET only) procedure FilterChanging(sender: System.Object; e:
AxEXPLORERTREELib._IExplorerTreeEvents_FilterChangingEvent);
begin
end;
```

```
PowerBuilder begin event FilterChanging(oleobject Group)
end event FilterChanging
```

```
VB.NET Private Sub FilterChanging(ByVal sender As System.Object, ByVal e As
AxEXPLORERTREELib._IExplorerTreeEvents_FilterChangingEvent) Handles
FilterChanging
End Sub
```

```
VB6 Private Sub FilterChanging(ByVal Group As EXPLORERTREELibCtl.IGroup)
End Sub
```

```
VBA Private Sub FilterChanging(ByVal Group As Object)
End Sub
```

```
VFP LPARAMETERS Group
```

```
Xbase... PROCEDURE OnFilterChanging(oExplorerTree,Group)
RETURN
```

Syntax for FilterChanging event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="FilterChanging(Group)" LANGUAGE="JScript">
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function FilterChanging(Group)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComFilterChanging Variant IIGroup  
    Forward Send OnComFilterChanging IIGroup  
End_Procedure
```

Visual
Objects

```
METHOD OCX_FilterChanging(Group) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_FilterChanging(COM _Group)  
{  
}
```

XBasic

```
function FilterChanging as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup)  
end function
```

dBASE

```
function nativeObject_FilterChanging(Group)  
return
```

event FormatColumn (Group as Group, Item as HITEM, ColIndex as Long, ByRef Value as Variant)

Fired when a cell requires to format its caption.

Type	Description
Group as Group	A Group object where the cell needs to be formatted
Item as HITEM	A long expression that indicates the handle of the item being formatted.
ColIndex as Long	A long expression that indicates the index of the column being formatted.
Value as Variant	(By Reference) A Variant value that indicates the value being formatted. By default, the Value parameter has the CellCaption value.

The FormatColumn event is fired only if the [FireFormatColumn](#) property of the Column object is True. The FormatColumn event lets the user provides the cell's caption before it being displayed on the group's list. For instance, the FormatColumn event is very useful when the column cells contains prices(numbers), and you want to display that column formatted as currency, like \$50 instead 50.

Syntax for FormatColumn event, **/NET** version, on:

```
C# private void FormatColumn(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,int ColIndex,ref object Value)
{
}
```

```
VB Private Sub FormatColumn(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal ColIndex As
Integer,ByRef Value As Object) Handles FormatColumn
End Sub
```

Syntax for FormatColumn event, **/COM** version, on:

```
C# private void FormatColumn(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_FormatColumnEvent e)
{
}
```

C++

```
void OnFormatColumn(LPDISPATCH Group,long Item,long ColIndex,VARIANT FAR* Value)
{
}
```

**C++
Builder**

```
void __fastcall FormatColumn(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::HITEM Item,long ColIndex,Variant * Value)
{
}
```

Delphi

```
procedure FormatColumn(ASender: TObject; Group : IGroup;Item :
HITEM;ColIndex : Integer;var Value : OleVariant);
begin
end;
```

**Delphi 8
(.NET
only)**

```
procedure FormatColumn(sender: System.Object; e:
AxEXPLORERTREELib._IExplorerTreeEvents_FormatColumnEvent);
begin
end;
```

Power...

```
begin event FormatColumn(oleobject Group,long Item,long ColIndex,any Value)
end event FormatColumn
```

VB.NET

```
Private Sub FormatColumn(ByVal sender As System.Object, ByVal e As
AxEXPLORERTREELib._IExplorerTreeEvents_FormatColumnEvent) Handles
FormatColumn
End Sub
```

VB6

```
Private Sub FormatColumn(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal
Item As EXPLORERTREELibCtl.HITEM,ByVal ColIndex As Long,Value As Variant)
End Sub
```

VBA

```
Private Sub FormatColumn(ByVal Group As Object,ByVal Item As Long,ByVal
ColIndex As Long,Value As Variant)
End Sub
```

VFP

```
LPARAMETERS Group,Item,ColIndex,Value
```

```
Xbas... PROCEDURE OnFormatColumn(oExplorerTree,Group,Item,ColIndex,Value)
RETURN
```

Syntax for FormatColumn event, **/COM** version (others), on:

```
Java... <SCRIPT EVENT="FormatColumn(Group,Item,ColIndex,Value)"
LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function FormatColumn(Group,Item,ColIndex,Value)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComFormatColumn Variant IIGroup HITEM IItem Integer IColIndex
Variant IValue
Forward Send OnComFormatColumn IIGroup IItem IColIndex IValue
End_Procedure
```

```
Visual Objects METHOD OCX_FormatColumn(Group,Item,ColIndex,Value) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_FormatColumn(COM _Group,int _Item,int _ColIndex,COMVariant
/*variant*/ _Value)
{
}
```

```
XBasic function FormatColumn as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item
as OLE::Exontrol.ExplorerTree.1::HITEM,ColIndex as N,Value as A)
end function
```

```
dBASE function nativeObject_FormatColumn(Group,Item,ColIndex,Value)
return
```

The following sample uses the FormatCurrency function provided by the VB to display numbers as currency:

```
Private Sub ExplorerTree1_FormatColumn(ByVal Group As EXPLOREERTREELibCtl.IGroup,  
ByVal Item As EXPLOREERTREELibCtl.HITEM, ByVal ColIndex As Long, Value As Variant)  
    Value = FormatCurrency(Value)  
End Sub
```

```
Private Sub Form_Load()  
    With ExplorerTree1  
        With .Groups.Add("Group 1")  
            .BeginUpdate  
                With .Columns(0)  
                    .FireFormatColumn = True  
                End With  
            .PutItems Array(10, 20, 30, 40)  
        .EndUpdate  
    End With  
End With  
End Sub
```

event HyperLinkClick (Group as Group, Item as HITEM, ColIndex as Long)

Occurs when the user clicks on a hyperlink cell.

Type	Description
Group as Group	A Group object where user clicks a hyper link cell.
Item as HITEM	A long expression that indicates the item's handle.
ColIndex as Long	A long expression that indicates the column's index.

The HyperLinkClick event is fired when user clicks a hyperlink cell. A hyperlink cell has the [CellHyperLink](#) property true. Use the HyperLinkClick event to notify your application that a hyperlink cell is clicked. Use the [HyperLinkColor](#) property to specify the color of the hyper links items.

Syntax for HyperLinkClick event, **/NET** version, on:

```
C# private void HyperLinkClick(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,int ColIndex)
{
}
```

```
VB Private Sub HyperLinkClick(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal ColIndex As Integer)
Handles HyperLinkClick
End Sub
```

Syntax for HyperLinkClick event, **/COM** version, on:

```
C# private void HyperLinkClick(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_HyperLinkClickEvent e)
{
}
```

```
C++ void OnHyperLinkClick(LPDISPATCH Group,long Item,long ColIndex)
{
}
```

```
C++ Builder void __fastcall HyperLinkClick(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::HITEM Item,long ColIndex)
```



```
{  
}
```

```
Delphi  
procedure HyperLinkClick(ASender: TObject; Group : IGroup;Item :  
HITEM;ColIndex : Integer);  
begin  
end;
```

```
Delphi 8  
(.NET  
only)  
procedure HyperLinkClick(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_HyperLinkClickEvent);  
begin  
end;
```

```
Powe...  
begin event HyperLinkClick(oleobject Group,long Item,long ColIndex)  
end event HyperLinkClick
```

```
VB.NET  
Private Sub HyperLinkClick(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_HyperLinkClickEvent) Handles  
HyperLinkClick  
End Sub
```

```
VB6  
Private Sub HyperLinkClick(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Item As EXPLORERTREELibCtl.HITEM,ByVal ColIndex As Long)  
End Sub
```

```
VBA  
Private Sub HyperLinkClick(ByVal Group As Object,ByVal Item As Long,ByVal  
ColIndex As Long)  
End Sub
```

```
VFP  
LPARAMETERS Group,Item,ColIndex
```

```
Xbas...  
PROCEDURE OnHyperLinkClick(oExplorerTree,Group,Item,ColIndex)  
RETURN
```

Syntax for HyperLinkClick event, **/COM** version (others), on:

```
Java...  
<SCRIPT EVENT="HyperLinkClick(Group,Item,ColIndex)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function HyperLinkClick(Group,Item,ColIndex)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComHyperLinkClick Variant IIGroup HITEM IItem Integer IIColIndex  
Forward Send OnComHyperLinkClick IIGroup IItem IIColIndex  
End_Procedure
```

Visual
Objects

```
METHOD OCX_HyperLinkClick(Group,Item,ColIndex) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_HyperLinkClick(COM _Group,int _Item,int _ColIndex)  
{  
}
```

XBasic

```
function HyperLinkClick as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item  
as OLE::Exontrol.ExplorerTree.1::HITEM,ColIndex as N)  
end function
```

dBASE

```
function nativeObject_HyperLinkClick(Group,Item,ColIndex)  
return
```

The following sample displays the caption of the clicked cell:

```
Private Sub ExplorerTree1_AddItem(ByVal Group As EXPLORERTREELibCtl.IGroup, ByVal  
Item As EXPLORERTREELibCtl.HITEM)  
Group.Items.CellHyperLink(Item, 0) = True  
End Sub
```

```
Private Sub ExplorerTree1_HyperLinkClick(ByVal Group As EXPLORERTREELibCtl.IGroup,  
ByVal Item As EXPLORERTREELibCtl.HITEM, ByVal ColIndex As Long)  
Debug.Print Group.Items.CellCaption(Item, ColIndex)  
End Sub
```

```
Private Sub Form_Load()
```

```
With ExplorerTree1
```

```
With .Groups.Add("Group 1")
```

```
    .SelForeColor = .HyperLinkColor
```

```
    .SelBackColor = .BackColorList
```

```
    .PutItems Array("https://www.exontrol.com", "https://www.exontrol.net")
```

```
End With
```

```
End With
```

```
End Sub
```

event ItemOleEvent (Group as Group, Item as HITEM, Ev as OleEvent)

Fired when an ActiveX control hosted by an item has fired an event.

Type	Description
Group as Group	A Group object
Item as HITEM	A long expression that indicates the handle of the item that hosts an ActiveX control.
Ev as OleEvent	An OleEvent object that contains information about the fired event.

The ExplorerTree control supports ActiveX hosting. The [InsertItemControl](#) method inserts an item that hosts an ActiveX control. The ItemOleEvent event notifies your application when a hosted ActiveX control fires an event.

Syntax for ItemOleEvent event, **/NET** version, on:

```
C# private void ItemOleEvent(object sender,exontrol.EXPLORERTREELib.Group
Group,int Item,exontrol.EXPLORERTREELib.OleEvent Ev)
{
}
```

```
VB Private Sub ItemOleEvent(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal Ev As
exontrol.EXPLORERTREELib.OleEvent) Handles ItemOleEvent
End Sub
```

Syntax for ItemOleEvent event, **/COM** version, on:

```
C# private void ItemOleEvent(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_ItemOleEventEvent e)
{
}
```

```
C++ void OnItemOleEvent(LPDISPATCH Group,long Item,LPDISPATCH Ev)
{
}
```

```
C++ Builder void __fastcall ItemOleEvent(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::HITEM Item,Explorertreelib_tlb::IOleEvent *Ev)
```

```
{  
}
```

```
Delphi  
procedure ItemOleEvent(ASender: TObject; Group : IGroup;Item : HITEM;Ev :  
IOleEvent);  
begin  
end;
```

```
Delphi 8  
(.NET  
only)  
procedure ItemOleEvent(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_ItemOleEventEvent);  
begin  
end;
```

```
Powe...  
begin event ItemOleEvent(oleobject Group,long Item,oleobject Ev)  
end event ItemOleEvent
```

```
VB.NET  
Private Sub ItemOleEvent(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_ItemOleEventEvent) Handles  
ItemOleEvent  
End Sub
```

```
VB6  
Private Sub ItemOleEvent(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal Item  
As EXPLORERTREELibCtl.HITEM,ByVal Ev As EXPLORERTREELibCtl.IOleEvent)  
End Sub
```

```
VBA  
Private Sub ItemOleEvent(ByVal Group As Object,ByVal Item As Long,ByVal Ev As  
Object)  
End Sub
```

```
VFP  
LPARAMETERS Group,Item,Ev
```

```
Xbas...  
PROCEDURE OnItemOleEvent(oExplorerTree,Group,Item,Ev)  
RETURN
```

Syntax for ItemOleEvent event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="ItemOleEvent(Group,Item,Ev)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function ItemOleEvent(Group,Item,Ev)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComItemOleEvent Variant IIGroup HITEM IItem Variant IIEv  
Forward Send OnComItemOleEvent IIGroup IItem IIEv  
End_Procedure
```

Visual
Objects

```
METHOD OCX_ItemOleEvent(Group,Item,Ev) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_ItemOleEvent(COM_Group,int_Item,COM_Ev)  
{  
}
```

XBasic

```
function ItemOleEvent as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item as  
OLE::Exontrol.ExplorerTree.1::HITEM,Ev as OLE::Exontrol.ExplorerTree.1::IOleEvent)  
end function
```

dBASE

```
function nativeObject_ItemOleEvent(Group,Item,Ev)  
return
```

The following sample adds an item that hosts the Microsoft Calendar Control and prints each event fired by this ActiveX control:

```
Private Sub ExplorerTree1_ItemOleEvent(ByVal Group As EXPLORERTREELibCtl.IGroup,  
ByVal Item As EXPLORERTREELibCtl.HITEM, ByVal Ev As EXPLORERTREELibCtl.IOleEvent)  
Debug.Print "Event name:" & Ev.Name  
If (Ev.CountParam = 0) Then  
Debug.Print "The event has no arguments."  
Else  
Debug.Print "The event has the following arguments:"  
Dim i As Long  
For i = 0 To Ev.CountParam - 1  
Debug.Print Ev(i).Name; " = " & Ev(i).Value
```

```
Next
End If
End Sub
```

```
Private Sub Form_Load()
    With ExplorerTree1
        With .Groups.Add("Group 1")
            .Height = 172
            .Expanded = True
            .Items.ItemHeight(.Items.InsertControlItem("MSCAL.Calendar")) = .Height
        End With
    End With
End Sub
```

The [ItemObject](#) property gets the ActiveX object hosted by an item that is inserted using the InsertControlItem method. The ItemObject property gets nothing if the item doesn't host an ActiveX control, or if inserting an ActiveX control failed).

event KeyDown (ByRef KeyCode as Integer, Shift as Integer)

Occurs when the user presses a key while an object has the focus.

Type	Description
KeyCode as Integer	(By Reference) An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and [KeyUp](#) event procedures if you need to respond to both the pressing and releasing of a key. You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the And operator with the shift argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

```
ShiftDown = (Shift And 1) > 0  
CtrlDown = (Shift And 2) > 0  
AltDown = (Shift And 4) > 0
```

In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

Syntax for KeyDown event, **/NET** version, on:

```
C# private void KeyDown(object sender,ref short KeyCode,short Shift)  
{  
}
```

```
VB Private Sub KeyDown(ByVal sender As System.Object,ByRef KeyCode As  
Short,ByVal Shift As Short) Handles KeyDown  
End Sub
```

Syntax for KeyDown event, **/COM** version, on:

```
C# private void KeyDownEvent(object sender,  
AxEXPLORERTREELib._IExplorerTreeEvents_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:  
AxEXPLORERTREELib._IExplorerTreeEvents_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  
AxEXPLORERTREELib._IExplorerTreeEvents_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(oExplorerTree,KeyCode,Shift)
```

Syntax for KeyDown event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function KeyDown(KeyCode,Shift)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComKeyDown Short IIKeyCode Short IIShift
Forward Send OnComKeyDown IIKeyCode IIShift
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 (ByRef KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	(By Reference) An integer that returns a standard numeric ANSI keycode.

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use [KeyDown](#) and [KeyUp](#) event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress does not indicate the physical state of the keyboard; instead, it passes a character. KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters.

Syntax for KeyPress event, **/.NET** version, on:

```
C# private void KeyPress(object sender,ref short KeyAscii)
{
}
```

```
VB Private Sub KeyPress(ByVal sender As System.Object,ByRef KeyAscii As Short)
Handles KeyPress
End Sub
```

Syntax for KeyPress event, **/.COM** version, on:

```
C# private void KeyPressEvent(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_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:  
AxEXPLORERTREELib._IExplorerTreeEvents_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  
AxEXPLORERTREELib._IExplorerTreeEvents_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(oExplorerTree,KeyAscii)  
RETURN
```

Syntax for KeyPress event, **ICOM** 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 IIKeyAscii  
    Forward Send OnComKeyPress IIKeyAscii  
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 (ByRef KeyCode as Integer, Shift as Integer)

Occurs when the user releases a key while an object has the focus.

Type	Description
KeyCode as Integer	(By Reference) An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the KeyUp event procedure to respond to the releasing of a key.

Syntax for KeyUp event, **/NET** version, on:

```
C# private void KeyUp(object sender,ref short KeyCode,short Shift)
{
}
```

```
VB Private Sub KeyUp(ByVal sender As System.Object,ByRef KeyCode As Short,ByVal
Shift As Short) Handles KeyUp
End Sub
```

Syntax for KeyUp event, **/COM** version, on:

```
C# private void KeyUpEvent(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_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:  
AxEXPLORERTREELib._IExplorerTreeEvents_KeyUpEvent);  
begin  
end;
```

```
PowerBuilder begin event KeyUp(integer KeyCode,integer Shift)  
end event KeyUp
```

```
VB.NET Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_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
```

```
Xbase... PROCEDURE OnKeyUp(oExplorerTree,KeyCode,Shift)  
RETURN
```

Syntax for KeyUp event, **ICOM** 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 IIKeyCode Short IIShift  
    Forward Send OnComKeyUp IIKeyCode IIShift  
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 **LayoutChanged** ()

Occurs when control's layout is changed.

Type

Description

Use the **LayoutChanged** event to notify your application that the control's content is changed. The **LayoutChanged** event is called if the control is resized, if a group is expanded or collapsed, if the control is scrolled, and so on.

Syntax for **LayoutChanged** event, **/NET** version, on:

```
C# private void LayoutChanged(object sender)
{
}
```

```
VB Private Sub LayoutChanged(ByVal sender As System.Object) Handles
LayoutChanged
End Sub
```

Syntax for **LayoutChanged** event, **/COM** version, on:

```
C# private void LayoutChanged(object sender, EventArgs e)
{
}
```

```
C++ void OnLayoutChanged()
{
}
```

```
C++ Builder void __fastcall LayoutChanged(TObject *Sender)
{
}
```

```
Delphi procedure LayoutChanged(ASender: TObject; );
begin
end;
```

```
Delphi 8 (.NET only) procedure LayoutChanged(sender: System.Object; e: System.EventArgs);
begin
end;
```

```
Powe... begin event LayoutChanged()  
end event LayoutChanged
```

```
VB.NET Private Sub LayoutChanged(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles LayoutChanged  
End Sub
```

```
VB6 Private Sub LayoutChanged()  
End Sub
```

```
VBA Private Sub LayoutChanged()  
End Sub
```

```
VFP LPARAMETERS nop
```

```
Xbas... PROCEDURE OnLayoutChanged(oExplorerTree)  
RETURN
```

Syntax for LayoutChanged event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="LayoutChanged()" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBSc... <SCRIPT LANGUAGE="VBScript">  
Function LayoutChanged()  
End Function  
</SCRIPT>
```

```
Visual  
Data... Procedure OnComLayoutChanged  
Forward Send OnComLayoutChanged  
End_Procedure
```

```
Visual  
Objects METHOD OCX_LayoutChanged() CLASS MainDialog  
RETURN NIL
```

```
X++ void onEvent_LayoutChanged()  
{
```

```
}
```

XBasic

```
function LayoutChanged as v ()  
end function
```

dBASE

```
function nativeObject_LayoutChanged()  
return
```

event MouseDown (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user presses a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

Use a MouseDown or [MouseDown](#) event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the [Click](#) and [DbClick](#) events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the [GroupFromPoint](#) property to get the group's caption from the point. Use the [GroupListFromPoint](#) property to get the group's list from point. Use the [ItemFromPoint](#) property to get the item over the cursor. Use the [ColumnFromPoint](#) property to get the column's caption over the cursor. The [ExpandOnClick](#) property expands the group when its caption is clicked. Use the [ShortcutFromPoint](#) property to retrieve the shortcut from the cursor.

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,
AxEXPLORERTREELib._IExplorerTreeEvents_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:
AxEXPLORERTREELib._IExplorerTreeEvents_MouseDownEvent);
begin
end;
```

```
Power... 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
AxEXPLORERTREELib._IExplorerTreeEvents_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(oExplorerTree,Button,Shift,X,Y)  
RETURN
```

Syntax for MouseDown event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">  
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function MouseDown(Button,Shift,X,Y)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComMouseDown Short lButton Short lShift OLE_XPOS_PIXELS lX  
OLE_YPOS_PIXELS lY  
    Forward Send OnComMouseDown lButton lShift lX lY  
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.ExplorerTree.1::OLE_XPOS_PIXELS,Y as  
OLE::Exontrol.ExplorerTree.1::OLE_YPOS_PIXELS)  
end function
```

dBASE

```
function nativeObject_MouseDown(Button,Shift,X,Y)  
return
```

The following VB sample prints the cell over the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
  Dim g As EXPLORERTREELibCtl.Group
  With ExplorerTree1
    Set g = .GroupListFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    If Not (g Is Nothing) Then
      With g
        Dim h As Long, c As Long, hit as Long
        h = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, hit)
        If Not (h = 0) Then
          With g.Items
            Debug.Print .CellCaption(h, c)
          End With
        End If
      End With
    End If
  End With
End Sub
```

The following VFP sample prints the cell's caption from the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform.Olecontrol1
  local g
  g = .GroupListFromPoint(-1,-1)
  If !isnull(g) Then
    with g
      local h, c, hit
      c = 0
      hit = 0
      h = .ItemFromPoint(-1,-1,@c,@hit)
      If h # 0 Then
        with .Items
          .DefaultItem = h
        end with
      end if
    end with
  end if
end with
```

```
wait window .CellCaption(0, c) nowait
```

```
endwith
```

```
EndIf
```

```
endwith
```

```
EndIf
```

```
endwith
```


event MouseEventArgs (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 MouseEventArgs event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseEventArgs event whenever the mouse position is within its borders. Use the [GroupFromPoint](#) property to get the group's caption from the point. Use the [GroupListFromPoint](#) property to get the group's list from point. Use the [ItemFromPoint](#) property to get the item over the cursor. Use the [ColumnFromPoint](#) property to get the column's caption over the cursor. Use the [ShortcutFromPoint](#) property to retrieve the shortcut from the cursor.

Syntax for MouseEventArgs event, **/NET** version, on:

```
C# private void MouseEventArgs(object sender,short Button,short Shift,int X,int Y)
{
}
```

```
VB Private Sub MouseEventArgs(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseEventArgs
End Sub
```

Syntax for MouseEventArgs event, **/COM** version, on:

```
C# private void MouseEventArgs(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_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: AxEXPLORERTREELib._IExplorerTreeEvents_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 AxEXPLORERTREELib._IExplorerTreeEvents_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(oExplorerTree,Button,Shift,X,Y)  
RETURN
```

Syntax for MouseMove event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="MouseMove(Button,Shift,X,Y)" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">  
Function MouseMove(Button,Shift,X,Y)  
End Function  
</SCRIPT>
```

```
Visual  
Data... Procedure OnComMouseMove Short IButton Short IShift OLE_XPOS_PIXELS IIX  
OLE_YPOS_PIXELS IY  
Forward Send OnComMouseMove IButton IShift IIX IY  
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.ExplorerTree.1::OLE_XPOS_PIXELS,Y as  
OLE::Exontrol.ExplorerTree.1::OLE_YPOS_PIXELS)  
end function
```

```
dBASE function nativeObject_MouseMove(Button,Shift,X,Y)  
return
```

The following VB sample prints the cell over the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
```

As Single)

```
Dim g As EXPLORERTREELibCtl.Group
```

```
With ExplorerTree1
```

```
Set g = .GroupListFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
```

```
If Not (g Is Nothing) Then
```

```
With g
```

```
Dim h As Long, c As Long, hit as Long
```

```
h = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, hit)
```

```
If Not (h = 0) Then
```

```
With g.Items
```

```
Debug.Print .CellCaption(h, c)
```

```
End With
```

```
End If
```

```
End With
```

```
End If
```

```
End With
```

```
End Sub
```

The following VFP sample prints the cell's caption from the cursor:

```
*** ActiveX Control Event ***
```

```
LPARAMETERS button, shift, x, y
```

```
with thisform.Olecontrol1
```

```
local g
```

```
g = .GroupListFromPoint(-1,-1)
```

```
If !isnull(g) Then
```

```
with g
```

```
local h, c, hit
```

```
c = 0
```

```
hit = 0
```

```
h = .ItemFromPoint(-1,-1,@c,@hit)
```

```
If h # 0 Then
```

```
with .Items
```

```
.DefaultItem = h
```

```
wait window .CellCaption(0, c) nowait
```

```
endwith
```

EndIf
endwith
EndIf
endwith

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 [DbClick](#) events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the [GroupFromPoint](#) property to get the group's caption from the point. Use the [GroupListFromPoint](#) property to get the group's list from point. Use the [ItemFromPoint](#) property to get the item over the cursor. Use the [ColumnFromPoint](#) property to get the column's caption over the cursor. The [ExpandOnClick](#) property expands the group when its caption is clicked. Use the [ShortcutFromPoint](#) property to retrieve the shortcut from the cursor.

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,
AxEXPLORERTREELib._IExplorerTreeEvents_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:
AxEXPLORERTREELib._IExplorerTreeEvents_MouseUpEvent);
begin
end;
```

```
Power... 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
AxEXPLORERTREELib._IExplorerTreeEvents_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(oExplorerTree,Button,Shift,X,Y)  
RETURN
```

Syntax for MouseUp event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="MouseUp(Button,Shift,X,Y)" LANGUAGE="JScript">  
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function MouseUp(Button,Shift,X,Y)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComMouseUp Short IButton Short IShift OLE_XPOS_PIXELS IIX  
OLE_YPOS_PIXELS IY  
    Forward Send OnComMouseUp IButton IShift IIX IY  
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.ExplorerTree.1::OLE_XPOS_PIXELS,Y as  
OLE::Exontrol.ExplorerTree.1::OLE_YPOS_PIXELS)  
end function
```

dBASE

```
function nativeObject_MouseUp(Button,Shift,X,Y)  
return
```


The following VB sample prints the cell over the cursor:

```
Private Sub ExplorerTree1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
  Dim g As EXPLORERTREELibCtl.Group
  With ExplorerTree1
    Set g = .GroupListFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    If Not (g Is Nothing) Then
      With g
        Dim h As Long, c As Long, hit as Long
        h = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, hit)
        If Not (h = 0) Then
          With g.Items
            Debug.Print .CellCaption(h, c)
          End With
        End If
      End With
    End If
  End With
End Sub
```

The following VFP sample prints the cell's caption from the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform.Olecontrol1
  local g
  g = .GroupListFromPoint(-1,-1)
  If !isnull(g) Then
    with g
      local h, c, hit
      c = 0
      hit = 0
      h = .ItemFromPoint(-1,-1,@c,@hit)
      If h # 0 Then
        with .Items
          .DefaultItem = h
        end with
      end if
    end with
  end if
end with
```

```
wait window .CellCaption(0, c) nowait
```

```
endwith
```

```
EndIf
```

```
endwith
```

```
EndIf
```

```
endwith
```

event OffsetChanged (Group as Group, Horizontal as Boolean, NewVal as Long)

Occurs when the scroll position is changed.

Type	Description
Group as Group	A Group object where the scroll position is changed.
Horizontal as Boolean	A boolean expression that indicates whether the horizontal scroll bar has changed.
NewVal as Long	A long value that indicates the new scroll bar value in pixels.

If the group has no scroll bars the OffsetChanged and [OversizeChanged](#) events are not fired. Use the [ScrollBars](#) property of the control to determine which scroll bars are visible within the control.

Syntax for OffsetChanged event, **/NET** version, on:

```
C# private void OffsetChanged(object sender,exontrol.EXPLORERTREELib.Group
Group,bool Horizontal,int NewVal)
{
}
```

```
VB Private Sub OffsetChanged(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Horizontal As Boolean,ByVal NewVal As
Integer) Handles OffsetChanged
End Sub
```

Syntax for OffsetChanged event, **/COM** version, on:

```
C# private void OffsetChanged(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_OffsetChangedEvent e)
{
}
```

```
C++ void OnOffsetChanged(LPDISPATCH Group,BOOL Horizontal,long NewVal)
{
}
```

C++
Builder

```
void __fastcall OffsetChanged(TObject *Sender,Explorertreelib_tlb::IGroup  
*Group,VARIANT_BOOL Horizontal,long NewVal)  
{  
}
```

Delphi

```
procedure OffsetChanged(ASender: TObject; Group : IGroup;Horizontal :  
WordBool;NewVal : Integer);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure OffsetChanged(sender: System.Object; e:  
AxEXPLOREERTREELib._IExplorerTreeEvents_OffsetChangedEvent);  
begin  
end;
```

Powe...

```
begin event OffsetChanged(oleobject Group,boolean Horizontal,long NewVal)  
end event OffsetChanged
```

VB.NET

```
Private Sub OffsetChanged(ByVal sender As System.Object, ByVal e As  
AxEXPLOREERTREELib._IExplorerTreeEvents_OffsetChangedEvent) Handles  
OffsetChanged  
End Sub
```

VB6

```
Private Sub OffsetChanged(ByVal Group As EXPLOREERTREELibCtl.IGroup,ByVal  
Horizontal As Boolean,ByVal NewVal As Long)  
End Sub
```

VBA

```
Private Sub OffsetChanged(ByVal Group As Object,ByVal Horizontal As  
Boolean,ByVal NewVal As Long)  
End Sub
```

VFP

```
LPARAMETERS Group,Horizontal,NewVal
```

Xbas...

```
PROCEDURE OnOffsetChanged(oExplorerTree,Group,Horizontal,NewVal)  
RETURN
```

Syntax for OffsetChanged event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="OffsetChanged(Group,Horizontal,NewVal)"
LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function OffsetChanged(Group,Horizontal,NewVal)
End Function
</SCRIPT>
```

```
Visual
Data... Procedure OnComOffsetChanged Variant IIGroup Boolean IIHorizontal Integer
IINewVal
Forward Send OnComOffsetChanged IIGroup IIHorizontal IINewVal
End_Procedure
```

```
Visual
Objects METHOD OCX_OffsetChanged(Group,Horizontal,NewVal) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_OffsetChanged(COM _Group,boolean _Horizontal,int _NewVal)
{
}
```

```
XBasic function OffsetChanged as v (Group as
OLE::Exontrol.ExplorerTree.1::IGroup,Horizontal as L,NewVal as N)
end function
```

```
dBASE function nativeObject_OffsetChanged(Group,Horizontal,NewVal)
return
```

The following sample displays the position of the scroll bar that user changes:

```
Private Sub ExplorerTree1_OffsetChanged(ByVal Group As EXPLOREERTREELibCtl.IGroup,
ByVal Horizontal As Boolean, ByVal NewVal As Long)
Debug.Print "The user changes the " & If(Horizontal, "horizontal", "vertical") & " scroll
bar position. The new position is " & NewVal
End Sub
```


event **OLECompleteDrag** (Effect as Long)

Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled

Type	Description
Effect as Long	A long set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another)

The **OLECompleteDrag** event is the final event to be called in an OLE drag/drop operation. This event informs the source component of the action that was performed when the object was dropped onto the target component. The target sets this value through the effect parameter of the [OLEDragDrop](#) event. Based on this, the source can then determine the appropriate action it needs to take. For example, if the object was moved into the target (`exDropEffectMove`), the source needs to delete the object from itself after the move. The control supports only manual OLE drag and drop events. In order to enable OLE drag and drop feature into control you have to set the [OLEDropMode](#) and [OLEDrag](#) properties.

The settings for Effect are:

- `exOLEDropEffectNone` (0), Drop target cannot accept the data, or the drop operation was cancelled
- `exOLEDropEffectCopy` (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- `exOLEDropEffectMove` (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Syntax for **OLECompleteDrag** event, **/NET** version, on:

```
C# // OLECompleteDrag event is not supported. Use the  
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

```
VB // OLECompleteDrag event is not supported. Use the  
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for **OLECompleteDrag** event, **/COM** version, on:

```
C# private void OLECompleteDrag(object sender,  
AxEXPLORERTREELib._IExplorerTreeEvents_OLECompleteDragEvent e)  
{
```

```
}
```

```
C++ void OnOLECompleteDrag(long Effect)
```

```
{  
}
```

```
C++ Builder void __fastcall OLECompleteDrag(TObject *Sender,long Effect)
```

```
{  
}
```

```
Delphi procedure OLECompleteDrag(ASender: TObject; Effect : Integer);  
begin  
end;
```

```
Delphi 8  
(.NET  
only) procedure OLECompleteDrag(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_OLECompleteDragEvent);  
begin  
end;
```

```
Powe... begin event OLECompleteDrag(long Effect)  
end event OLECompleteDrag
```

```
VB.NET Private Sub OLECompleteDrag(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_OLECompleteDragEvent) Handles  
OLECompleteDrag  
End Sub
```

```
VB6 Private Sub OLECompleteDrag(ByVal Effect As Long)  
End Sub
```

```
VBA Private Sub OLECompleteDrag(ByVal Effect As Long)  
End Sub
```

```
VFP LPARAMETERS Effect
```

```
Xbas... PROCEDURE OnOLECompleteDrag(oExplorerTree,Effect)  
RETURN
```


Syntax for OLECompleteDrag event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="OLECompleteDrag(Effect)" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">  
Function OLECompleteDrag(Effect)  
End Function  
</SCRIPT>
```

```
Visual  
Data... Procedure OnComOLECompleteDrag Integer IEffect  
Forward Send OnComOLECompleteDrag IEffect  
End_Procedure
```

```
Visual  
Objects METHOD OCX_OLECompleteDrag(Effect) CLASS MainDialog  
RETURN NIL
```

```
X++ // OLECompleteDrag event is not supported. Use the  
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

```
XBasic function OLECompleteDrag as v (Effect as N)  
end function
```

```
dBASE function nativeObject_OLECompleteDrag(Effect)  
return
```

event **OLEDragDrop** (Data as **ExDataObject**, Effect as **Long**, Button as **Integer**, Shift as **Integer**, X as **OLE_XPOS_PIXELS**, Y as **OLE_YPOS_PIXELS**)

Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject , it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here.
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks .
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT , CTRL , and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
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.

The OLEDragDrop event is fired when the user has dropped files or clipboard information into the control. Use the [OLEDropMode](#) property on exOLEDropManual to enable OLE drop and drop support. Use the [ItemFromPoint](#) property to get the item from point. Use the [ColumnFromPoint](#) property to get the column from point. Use the [AddItem](#) method to add a new item to the control. Use the [InsertItem](#) method to insert a new child item. Use the [ItemPosition](#) property to specify the item's position.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Syntax for OLEDragDrop event, **/NET** version, on:

```
C# // OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

```
VB // OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

Syntax for OLEDragDrop event, **/COM** version, on:

```
C# private void OLEDragDrop(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_OLEDragDropEvent e)
{
}
```

```
C++ void OnOLEDragDrop(LPDISPATCH Data,long FAR* Effect,short Button,short
Shift,long X,long Y)
{
}
```

```
C++ Builder void __fastcall OLEDragDrop(TObject *Sender,Explorertreelib_tlb::IExDataObject
*Data,long * Effect,short Button,short Shift,int X,int Y)
{
}
```

```
Delphi procedure OLEDragDrop(ASender: TObject; Data : IExDataObject;var Effect : Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer);  
begin  
end;
```

```
Delphi 8  
(.NET  
only) procedure OLEDragDrop(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_OLEDragDropEvent);  
begin  
end;
```

```
Powe... begin event OLEDragDrop(oleobject Data,long Effect,integer Button,integer  
Shift,long X,long Y)  
end event OLEDragDrop
```

```
VB.NET Private Sub OLEDragDrop(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_OLEDragDropEvent) Handles  
OLEDragDrop  
End Sub
```

```
VB6 Private Sub OLEDragDrop(ByVal Data As  
EXPLORERTREELibCtl.IExDataObject,Effect As Long,ByVal Button As Integer,ByVal  
Shift As Integer,ByVal X As Single,ByVal Y As Single)  
End Sub
```

```
VBA Private Sub OLEDragDrop(ByVal Data As Object,Effect As Long,ByVal Button As  
Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)  
End Sub
```

```
VFP LPARAMETERS Data,Effect,Button,Shift,X,Y
```

```
Xbas... PROCEDURE OnOLEDragDrop(oExplorerTree,Data,Effect,Button,Shift,X,Y)  
RETURN
```

Syntax for OLEDragDrop event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="OLEDragDrop(Data,Effect,Button,Shift,X,Y)"  
LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function OLEDragDrop(Data,Effect,Button,Shift,X,Y)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComOLEDragDrop Variant IIData Integer IIEffect Short IIButton  
Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS IIY  
    Forward Send OnComOLEDragDrop IIData IIEffect IIButton IIShift IIX IIY  
End_Procedure
```

Visual
Objects

```
METHOD OCX_OLEDragDrop(Data,Effect,Button,Shift,X,Y) CLASS MainDialog  
RETURN NIL
```

X++

```
// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver,  
DragDrop ... events.
```

XBasic

```
function OLEDragDrop as v (Data as  
OLE::Exontrol.ExplorerTree.1::IExDataObject,Effect as N,Button as N,Shift as N,X as  
OLE::Exontrol.ExplorerTree.1::OLE_XPOS_PIXELS,Y as  
OLE::Exontrol.ExplorerTree.1::OLE_YPOS_PIXELS)  
end function
```

dBASE

```
function nativeObject_OLEDragDrop(Data,Effect,Button,Shift,X,Y)  
return
```

event **OLEDragOver** (Data as **ExDataObject**, Effect as **Long**, Button as **Integer**, Shift as **Integer**, X as **OLE_XPOS_PIXELS**, Y as **OLE_YPOS_PIXELS**, State as **Integer**)

Occurs when one component is dragged over another.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject , it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks .
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT , CTRL , and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
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.

State as Integer

An integer that corresponds to the transition state of the control being dragged in relation to a target form or control. The possible values are listed in Remarks.

The settings for effect are:

- `exOLEDropEffectNone` (0), Drop target cannot accept the data, or the drop operation was cancelled
- `exOLEDropEffectCopy` (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- `exOLEDropEffectMove` (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The settings for state are:

- `exOLEDragEnter` (0), Source component is being dragged within the range of a target.
- `exOLEDragLeave` (1), Source component is being dragged out of the range of a target.
- `exOLEOLEDragOver` (2), Source component has moved from one position in the target to another.

Note If the state parameter is 1, indicating that the mouse pointer has left the target, then the x and y parameters will contain zeros.

The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an effect against, say, `exOLEDropEffectCopy`, such as in this manner:

If `Effect = exOLEDropEffectCopy...`

Instead, the source component should mask for the value or values being sought, such as this:

If `Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...`

-or-

If `(Effect And exOLEDropEffectCopy)...`

This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.

The control supports only manual OLE drag and drop events.

Syntax for `OLEDragOver` event, **/.NET** version, on:

```
C# // OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver,  
DragDrop ... events.
```

VB

```
// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for OLEDragOver event, **ICOM** version, on:

C#

```
private void OLEDragOver(object sender, AxEXPLORERTREELib._IExplorerTreeEvents_OLEDragOverEvent e)
{
}
```

C++

```
void OnOLEDragOver(LPDISPATCH Data,long FAR* Effect,short Button,short Shift,long X,long Y,short State)
{
}
```

C++**Builder**

```
void __fastcall OLEDragOver(TObject *Sender,Explorertreelib_tlb::IExDataObject *Data,long * Effect,short Button,short Shift,int X,int Y,short State)
{
}
```

Delphi

```
procedure OLEDragOver(ASender: TObject; Data : IExDataObject;var Effect : Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer;State : Smallint);
begin
end;
```

**Delphi 8
(.NET
only)**

```
procedure OLEDragOver(sender: System.Object; e: AxEXPLORERTREELib._IExplorerTreeEvents_OLEDragOverEvent);
begin
end;
```

Power...

```
begin event OLEDragOver(oleobject Data,long Effect,integer Button,integer Shift,long X,long Y,integer State)
end event OLEDragOver
```

VB.NET

```
Private Sub OLEDragOver(ByVal sender As System.Object, ByVal e As AxEXPLORERTREELib._IExplorerTreeEvents_OLEDragOverEvent) Handles OLEDragOver
End Sub
```



```
VB6 Private Sub OLEDragOver(ByVal Data As EXPLORERTREELibCtl.IExDataObject,Effect As Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Single,ByVal Y As Single,ByVal State As Integer)
End Sub
```

```
VBA Private Sub OLEDragOver(ByVal Data As Object,Effect As Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long,ByVal State As Integer)
End Sub
```

```
VFP LPARAMETERS Data,Effect,Button,Shift,X,Y,State
```

```
Xbas... PROCEDURE OnOLEDragOver(oExplorerTree,Data,Effect,Button,Shift,X,Y,State)
RETURN
```

Syntax for OLEDragOver event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="OLEDragOver(Data,Effect,Button,Shift,X,Y,State)"
LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function OLEDragOver(Data,Effect,Button,Shift,X,Y,State)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComOLEDragOver Variant IIData Integer IIEffect Short IIButton Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS IIY Short IIShift
Forward Send OnComOLEDragOver IIData IIEffect IIButton IIShift IIX IIY IIShift
End_Procedure
```

```
Visual Objects METHOD OCX_OLEDragOver(Data,Effect,Button,Shift,X,Y,State) CLASS MainDialog
RETURN NIL
```

```
X++ // OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

XBasic

```
function OLEDragOver as v (Data as  
OLE::Exontrol.ExplorerTree.1::IExDataObject,Effect as N,Button as N,Shift as N,X as  
OLE::Exontrol.ExplorerTree.1::OLE_XPOS_PIXELS,Y as  
OLE::Exontrol.ExplorerTree.1::OLE_YPOS_PIXELS,State as N)  
end function
```

dBASE

```
function nativeObject_OLEDragOver(Data,Effect,Button,Shift,X,Y,State)  
return
```

event OLEGiveFeedback (Effect as Long, DefaultCursors as Boolean)

Allows the drag source to specify the type of OLE drag-and-drop operation and the visual feedback.

Type	Description
Effect as Long	A long integer set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback). The possible values are listed in Remarks.
DefaultCursors as Boolean	Boolean value that determines whether to use the default mouse cursor, or to use a user-defined mouse cursor. True (default) = use default mouse cursor. False = do not use default cursor. Mouse cursor must be set with the MousePointer property of the Screen object.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

If there is no code in the OLEGiveFeedback event, or if the defaultcursors parameter is set to True, the mouse cursor will be set to the default cursor provided by the control. The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an effect against, say, exOLEDropEffectCopy, such as in this manner:

If Effect = exOLEDropEffectCopy...

Instead, the source component should mask for the value or values being sought, such as this:

If Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...

-or-

If (Effect And exOLEDropEffectCopy)...

This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.

The control supports only manual OLE drag and drop events.

Syntax for OLEGiveFeedback event, **/NET** version, on:

```
C# // OLEGiveFeedback event is not supported. Use the  
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

```
VB // OLEGiveFeedback event is not supported. Use the  
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for OLEGiveFeedback event, **/COM** version, on:

```
C# private void OLEGiveFeedback(object sender,  
AxEXPLORERTREELib._IExplorerTreeEvents_OLEGiveFeedbackEvent e)  
{  
}
```

```
C++ void OnOLEGiveFeedback(long Effect,BOOL FAR* DefaultCursors)  
{  
}
```

```
C++ Builder void __fastcall OLEGiveFeedback(TObject *Sender,long Effect,VARIANT_BOOL *  
DefaultCursors)  
{  
}
```

```
Delphi procedure OLEGiveFeedback(ASender: TObject; Effect : Integer;var DefaultCursors  
: WordBool);  
begin  
end;
```

```
Delphi 8  
(.NET  
only) procedure OLEGiveFeedback(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_OLEGiveFeedbackEvent);  
begin  
end;
```

```
Powe... begin event OLEGiveFeedback(long Effect,boolean DefaultCursors)  
end event OLEGiveFeedback
```

VB.NET

```
Private Sub OLEGiveFeedback(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_OLEGiveFeedbackEvent) Handles  
OLEGiveFeedback  
End Sub
```

VB6

```
Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean)  
End Sub
```

VBA

```
Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean)  
End Sub
```

VFP

```
LPARAMETERS Effect,DefaultCursors
```

Xbas...

```
PROCEDURE OnOLEGiveFeedback(oExplorerTree,Effect,DefaultCursors)  
RETURN
```

Syntax for OLEGiveFeedback event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="OLEGiveFeedback(Effect,DefaultCursors)"  
LANGUAGE="JScript">  
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function OLEGiveFeedback(Effect,DefaultCursors)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComOLEGiveFeedback Integer IEffect Boolean IIDefaultCursors  
Forward Send OnComOLEGiveFeedback IEffect IIDefaultCursors  
End_Procedure
```

Visual
Objects

```
METHOD OCX_OLEGiveFeedback(Effect,DefaultCursors) CLASS MainDialog  
RETURN NIL
```

X++

```
// OLEGiveFeedback event is not supported. Use the  
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

XBasic

```
function OLEGiveFeedback as v (Effect as N,DefaultCursors as L)
end function
```

dBASE

```
function nativeObject_OLEGiveFeedback(Effect,DefaultCursors)
return
```

event **OLESetData** (Data as **ExDataObject**, Format as **Integer**)

Occurs on a drag source when a drop target calls the `GetData` method and there is no data in a specified format in the OLE drag-and-drop `DataObject`.

Type	Description
Data as ExDataObject	An <code>ExDataObject</code> object in which to place the requested data. The component calls the <code>SetData</code> method to load the requested format.
Format as Integer	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the <code>ExDataObject</code> object.

The `OLESetData` is not currently supported.

Syntax for `OLESetData` event, **/NET** version, on:

```
C# // OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver,  
DragDrop ... events.
```

```
VB // OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver,  
DragDrop ... events.
```

Syntax for `OLESetData` event, **/COM** version, on:

```
C# private void OLESetData(object sender,  
AxEXPLORERTREELib._IExplorerTreeEvents_OLESetDataEvent e)  
{  
}
```

```
C++ void OnOLESetData(LPDISPATCH Data,short Format)  
{  
}
```

```
C++ Builder void __fastcall OLESetData(TObject *Sender,Explorertreelib_tlb::IExDataObject  
*Data,short Format)  
{  
}
```

Delphi

```
procedure OLESetData(ASender: TObject; Data : IExDataObject;Format : Smallint);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure OLESetData(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_OLESetDataEvent);  
begin  
end;
```

Power...

```
begin event OLESetData(oleobject Data,integer Format)  
end event OLESetData
```

VB.NET

```
Private Sub OLESetData(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_OLESetDataEvent) Handles OLESetData  
End Sub
```

VB6

```
Private Sub OLESetData(ByVal Data As EXPLORERTREELibCtl.IExDataObject,ByVal  
Format As Integer)  
End Sub
```

VBA

```
Private Sub OLESetData(ByVal Data As Object,ByVal Format As Integer)  
End Sub
```

VFP

```
LPARAMETERS Data,Format
```

Xbas...

```
PROCEDURE OnOLESetData(oExplorerTree,Data,Format)  
RETURN
```

Syntax for OLESetData event, **/COM** version (others), on:

Java...

```
<SCRIPT EVENT="OLESetData(Data,Format)" LANGUAGE="JScript">  
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function OLESetData(Data,Format)  
End Function
```



```
</SCRIPT>
```

Visual
Data...

```
Procedure OnComOLESetData Variant IIData Short IIFormat  
    Forward Send OnComOLESetData IIData IIFormat  
End_Procedure
```

Visual
Objects

```
METHOD OCX_OLESetData(Data,Format) CLASS MainDialog  
RETURN NIL
```

X++

```
// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver,  
DragDrop ... events.
```

XBasic

```
function OLESetData as v (Data as  
OLE::Exontrol.ExplorerTree.1::IExDataObject,Format as N)  
end function
```

dBASE

```
function nativeObject_OLESetData(Data,Format)  
return
```

event **OLEStartDrag** (Data as **ExDataObject**, AllowedEffects as **Long**)

Occurs when the **OLEDrag** method is called.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, optionally, the data for those formats. If no data is contained in the ExDataObject , it is provided when the control calls the GetData method. The programmer should provide the values for this parameter in this event. The SetData and Clear methods cannot be used here.
AllowedEffects as Long	A long containing the effects that the source component supports. The possible values are listed in Settings . The programmer should provide the values for this parameter in this event

The settings for **AllowEffects** are:

- **exOLEDropEffectNone** (0), Drop target cannot accept the data, or the drop operation was cancelled
- **exOLEDropEffectCopy** (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- **exOLEDropEffectMove** (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The source component should logically Or together the supported values and places the result in the **AllowedEffects** parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be). You may wish to defer putting data into the **ExDataObject** object until the target component requests it. This allows the source component to save time. If the user does not load any formats into the **ExDataObject**, then the drag/drop operation is canceled. Use [exCFFiles](#) and [Files](#) property to add files to the drag and drop data object.

The idea of drag and drop in **exExplorerTree** control is the same as in other controls. To start accepting drag and drop sources the **exExplorerTree** control should have the [OLEDropMode](#) to **exOLEDropManual**. Once that is set, the **exExplorerTree** starts accepting any drag and drop sources.

The first step is if you want to be able to drag items from your **exExplorerTree** control to other controls the idea is to handle the **OLE_StartDrag** event. The event passes an object **ExDataObject** (Data) as argument. The Data and **AllowedEffects** can be changed only in the **OLEStartDrag** event. The **OLE_StartDrag** event is fired when user is about to drag

items from the control. The **AllowedEffect** parameter and **SetData** property must be set to continue drag and drop operation, else the drag and drop operation is not started. Use the **FocusGroup** property determines the group where the drag and drop operations beings. Use the **FocusItem** property to determine the item that has the focus in the group. Use the **CellCaption** property to get the caption of the cell.

Syntax for OLEStartDrag event, **/NET** version, on:

```
C# // OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

```
VB // OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

Syntax for OLEStartDrag event, **/COM** version, on:

```
C# private void OLEStartDrag(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_OLEStartDragEvent e)
{
}
```

```
C++ void OnOLEStartDrag(LPDISPATCH Data,long FAR* AllowedEffects)
{
}
```

```
C++ Builder void __fastcall OLEStartDrag(TObject *Sender,Explorertreelib_tlb::IExDataObject
*Data,long * AllowedEffects)
{
}
```

```
Delphi procedure OLEStartDrag(ASender: TObject; Data : IExDataObject;var
AllowedEffects : Integer);
begin
end;
```

```
Delphi 8 (.NET only) procedure OLEStartDrag(sender: System.Object; e:
AxEXPLORERTREELib._IExplorerTreeEvents_OLEStartDragEvent);
begin
end;
```

Power... begin event OLEStartDrag(oleobject Data,long AllowedEffects)
end event OLEStartDrag

VB.NET Private Sub OLEStartDrag(ByVal sender As System.Object, ByVal e As AxEXPLORERTREELib._IExplorerTreeEvents_OLEStartDragEvent) Handles OLEStartDrag
End Sub

VB6 Private Sub OLEStartDrag(ByVal Data As EXPLORERTREELibCtl.IExDataObject,AllowedEffects As Long)
End Sub

VBA Private Sub OLEStartDrag(ByVal Data As Object,AllowedEffects As Long)
End Sub

VFP LPARAMETERS Data,AllowedEffects

Xbas... PROCEDURE OnOLEStartDrag(oExplorerTree,Data,AllowedEffects)
RETURN

Syntax for OLEStartDrag event, **COM** version (others), on:

Java... <SCRIPT EVENT="OLEStartDrag(Data,AllowedEffects)" LANGUAGE="JScript">
</SCRIPT>

VBSc... <SCRIPT LANGUAGE="VBScript">
Function OLEStartDrag(Data,AllowedEffects)
End Function
</SCRIPT>

Visual Data... Procedure OnComOLEStartDrag Variant IData Integer IAllowedEffects
Forward Send OnComOLEStartDrag IData IAllowedEffects
End_Procedure

Visual Objects METHOD OCX_OLEStartDrag(Data,AllowedEffects) CLASS MainDialog
RETURN NIL

X++

```
// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver,  
DragDrop ... events.
```

XBasic

```
function OLEStartDrag as v (Data as  
OLE::Exontrol.ExplorerTree.1::IExDataObject,AllowedEffects as N)  
end function
```

dBASE

```
function nativeObject_OLEStartDrag(Data,AllowedEffects)  
return
```

event **OversizeChanged** (Group as Group, Horizontal as Boolean, NewVal as Long)

Occurs when the right range of the scroll is changed.

Type	Description
Group as Group	A Group object where change occurs.
Horizontal as Boolean	A boolean expression that indicates whether the horizontal scroll bar has changed.
NewVal as Long	A long value that indicates the new scroll bar value.

If the control has no scroll bars the [OffsetChanged](#) and **OversizeChanged** events are not fired. When the scroll bar range is changed the **OversizeChanged** event is fired. Use the [ScrollBars](#) property of the control to determine which scroll bars are visible within the control.

Syntax for **OversizeChanged** event, **/NET** version, on:

```
C# private void OversizeChanged(object sender,exontrol.EXPLORERTREELib.Group
Group,bool Horizontal,int NewVal)
{
}
```

```
VB Private Sub OversizeChanged(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Horizontal As Boolean,ByVal NewVal As
Integer) Handles OversizeChanged
End Sub
```

Syntax for **OversizeChanged** event, **/COM** version, on:

```
C# private void OversizeChanged(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_OversizeChangedEvent e)
{
}
```

```
C++ void OnOversizeChanged(LPDISPATCH Group,BOOL Horizontal,long NewVal)
{
}
```

C++
Builder

```
void __fastcall OversizeChanged(TObject *Sender,ExplorerTreeLib_tlb::IGroup  
*Group,VARIANT_BOOL Horizontal,long NewVal)  
{  
}
```

Delphi

```
procedure OversizeChanged(ASender: TObject; Group : IGroup;Horizontal :  
WordBool;NewVal : Integer);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure OversizeChanged(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_OversizeChangedEvent);  
begin  
end;
```

Power...

```
begin event OversizeChanged(oleobject Group,boolean Horizontal,long NewVal)  
end event OversizeChanged
```

VB.NET

```
Private Sub OversizeChanged(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_OversizeChangedEvent) Handles  
OversizeChanged  
End Sub
```

VB6

```
Private Sub OversizeChanged(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Horizontal As Boolean,ByVal NewVal As Long)  
End Sub
```

VBA

```
Private Sub OversizeChanged(ByVal Group As Object,ByVal Horizontal As  
Boolean,ByVal NewVal As Long)  
End Sub
```

VFP

```
LPARAMETERS Group,Horizontal,NewVal
```

Xbas...

```
PROCEDURE OnOversizeChanged(oExplorerTree,Group,Horizontal,NewVal)  
RETURN
```

Syntax for OversizeChanged event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="OversizeChanged(Group,Horizontal,NewVal)"
LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function OversizeChanged(Group,Horizontal,NewVal)
End Function
</SCRIPT>
```

```
Visual
Data... Procedure OnComOversizeChanged Variant IIGroup Boolean IIHorizontal Integer
IINewVal
Forward Send OnComOversizeChanged IIGroup IIHorizontal IINewVal
End_Procedure
```

```
Visual
Objects METHOD OCX_OversizeChanged(Group,Horizontal,NewVal) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_OversizeChanged(COM _Group,boolean _Horizontal,int _NewVal)
{
}
```

```
XBasic function OversizeChanged as v (Group as
OLE::Exontrol.ExplorerTree.1::IGroup,Horizontal as L,NewVal as N)
end function
```

```
dBASE function nativeObject_OversizeChanged(Group,Horizontal,NewVal)
return
```


event RClick ()

Fired when right mouse button is clicked

Type

Description

The RClick event is fired each time the user releases the right mouse button over the control. Use the [MouseDown](#) or [MouseUp](#) event if you need the cursor coordinates. Else, you can use the GetCursorPos API function.

Syntax for RClick event, **/NET** version, on:

```
C# private void RClick(object sender)
{
}
```

```
VB Private Sub RClick(ByVal sender As System.Object) Handles RClick
End Sub
```

Syntax for RClick event, **/COM** version, on:

```
C# private void RClick(object sender, EventArgs e)
{
}
```

```
C++ void OnRClick()
{
}
```

```
C++ Builder void __fastcall RClick(TObject *Sender)
{
}
```

```
Delphi procedure RClick(ASender: TObject; );
begin
end;
```

```
Delphi 8 (.NET only) procedure RClick(sender: System.Object; e: System.EventArgs);
begin
end;
```

```
Powe... begin event RClick()  
end event RClick
```

```
VB.NET Private Sub RClick(ByVal sender As System.Object, ByVal e As System.EventArgs)  
Handles RClick  
End Sub
```

```
VB6 Private Sub RClick()  
End Sub
```

```
VBA Private Sub RClick()  
End Sub
```

```
VFP LPARAMETERS nop
```

```
Xbas... PROCEDURE OnRClick(oExplorerTree)  
RETURN
```

Syntax for RClick event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="RClick()" LANGUAGE="JScript">  
</SCRIPT>
```

```
VBSc... <SCRIPT LANGUAGE="VBScript">  
Function RClick()  
End Function  
</SCRIPT>
```

```
Visual  
Data... Procedure OnComRClick  
Forward Send OnComRClick  
End_Procedure
```

```
Visual  
Objects METHOD OCX_RClick() CLASS MainDialog  
RETURN NIL
```

```
X++ void onEvent_RClick()  
{
```

```
}
```

XBasic

```
function RClick as v ()  
end function
```

dBASE

```
function nativeObject_RClick()  
return
```

event RemoveColumn (Group as Group, Column as Column)

Fired before deleting a Column.

Type	Description
Group as Group	A Group object where a column is removed.
Column as Column	A Column object being removed.

The RemoveColumn event is invoked when the control is about to remove a column. Use the RemoveColumn event to release any extra data associated to the column. Use the [Remove](#) method to remove a specific column from Columns collection.

Syntax for RemoveColumn event, **/NET** version, on:

```
C# private void RemoveColumn(object sender,exontrol.EXPLORERTREELib.Group
Group,exontrol.EXPLORERTREELib.Column Column)
{
}
```

```
VB Private Sub RemoveColumn(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal Column As
exontrol.EXPLORERTREELib.Column) Handles RemoveColumn
End Sub
```

Syntax for RemoveColumn event, **/COM** version, on:

```
C# private void RemoveColumn(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveColumnEvent e)
{
}
```

```
C++ void OnRemoveColumn(LPDISPATCH Group,LPDISPATCH Column)
{
}
```

```
C++ Builder void __fastcall RemoveColumn(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::IColumn *Column)
{
}
```

Delphi

```
procedure RemoveColumn(ASender: TObject; Group : IGroup;Column : IColumn);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure RemoveColumn(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveColumnEvent);  
begin  
end;
```

Powe...

```
begin event RemoveColumn(oleobject Group,oleobject Column)  
end event RemoveColumn
```

VB.NET

```
Private Sub RemoveColumn(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveColumnEvent) Handles  
RemoveColumn  
End Sub
```

VB6

```
Private Sub RemoveColumn(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal  
Column As EXPLORERTREELibCtl.IColumn)  
End Sub
```

VBA

```
Private Sub RemoveColumn(ByVal Group As Object,ByVal Column As Object)  
End Sub
```

VFP

```
LPARAMETERS Group,Column
```

Xbas...

```
PROCEDURE OnRemoveColumn(oExplorerTree,Group,Column)  
RETURN
```

Syntax for RemoveColumn event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="RemoveColumn(Group,Column)" LANGUAGE="JScript">  
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function RemoveColumn(Group,Column)
```

```
End Function  
</SCRIPT>
```

```
Visual  
Data... Procedure OnComRemoveColumn Variant IIGroup Variant IIColumn  
Forward Send OnComRemoveColumn IIGroup IIColumn  
End_Procedure
```

```
Visual  
Objects METHOD OCX_RemoveColumn(Group,Column) CLASS MainDialog  
RETURN NIL
```

```
X++ void onEvent_RemoveColumn(COM _Group,COM _Column)  
{  
}
```

```
XBasic function RemoveColumn as v (Group as  
OLE::Exontrol.ExplorerTree.1::IGroup,Column as  
OLE::Exontrol.ExplorerTree.1::IColumn)  
end function
```

```
dBASE function nativeObject_RemoveColumn(Group,Column)  
return
```

event RemoveGroup (Group as Group)

Fired when a group is removed.

Type	Description
Group as Group	A Group object being removed.

Use the RemoveGroup event to notify your application that a group is released. Use the RemoveGroup event to release any extra data stored by the group. The [Remove](#) method fires the RemoveGroup event for each group removed. Use the RemoveItem event to notify your application that an item is deleted.

Syntax for RemoveGroup event, **/NET** version, on:

```
C# private void RemoveGroup(object sender,exontrol.EXPLORERTREELib.Group
Group)
{
}
```

```
VB Private Sub RemoveGroup(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group) Handles RemoveGroup
End Sub
```

Syntax for RemoveGroup event, **/COM** version, on:

```
C# private void RemoveGroup(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveGroupEvent e)
{
}
```

```
C++ void OnRemoveGroup(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall RemoveGroup(TObject *Sender,Explorertreelib_tlb::IGroup *Group)
{
}
```

```
Delphi procedure RemoveGroup(ASender: TObject; Group : IGroup);
begin
```

```
end;
```

Delphi 8
(.NET
only)

```
procedure RemoveGroup(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveGroupEvent);  
begin  
end;
```

Powe...

```
begin event RemoveGroup(oleobject Group)  
end event RemoveGroup
```

VB.NET

```
Private Sub RemoveGroup(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveGroupEvent) Handles  
RemoveGroup  
End Sub
```

VB6

```
Private Sub RemoveGroup(ByVal Group As EXPLORERTREELibCtl.IGroup)  
End Sub
```

VBA

```
Private Sub RemoveGroup(ByVal Group As Object)  
End Sub
```

VFP

```
LPARAMETERS Group
```

Xbas...

```
PROCEDURE OnRemoveGroup(oExplorerTree,Group)  
RETURN
```

Syntax for RemoveGroup event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="RemoveGroup(Group)" LANGUAGE="JScript">  
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function RemoveGroup(Group)  
End Function  
</SCRIPT>
```


Visual
Data...

```
Procedure OnComRemoveGroup Variant IIGroup  
    Forward Send OnComRemoveGroup IIGroup  
End_Procedure
```

Visual
Objects

```
METHOD OCX_RemoveGroup(Group) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_RemoveGroup(COM _Group)  
{  
}
```

XBasic

```
function RemoveGroup as v (Group as OLE::Exontrol.ExplorerTree.1::IIGroup)  
end function
```

dBASE

```
function nativeObject_RemoveGroup(Group)  
return
```

The following sample displays the group's caption being deleted:

```
Private Sub ExplorerTree1_RemoveGroup(ByVal Group As EXPLORERTREELibCtl.IIGroup)  
    Debug.Print Group.Caption  
End Sub
```

event RemoveItem (Group as Group, Item as HITEM)

Occurs before deleting an Item.

Type	Description
Group as Group	A Group object where the item is removed
Item as HITEM	A long expression that indicates the handle of the item being removed.

Use the RemoveItem to release any extra data that you might have used. The control fires the RemoveItem event before removing the item. Use the [RemoveItem](#) method to remove an item from Items collection.

Syntax for RemoveItem event, **/NET** version, on:

```
C# private void RemoveItem(object sender, exontrol.EXPLORERTREELib.Group
Group, int Item)
{
}
```

```
VB Private Sub RemoveItem(ByVal sender As System.Object, ByVal Group As
exontrol.EXPLORERTREELib.Group, ByVal Item As Integer) Handles RemoveItem
End Sub
```

Syntax for RemoveItem event, **/COM** version, on:

```
C# private void RemoveItem(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveItemEvent e)
{
}
```

```
C++ void OnRemoveItem(LPDISPATCH Group, long Item)
{
}
```

```
C++ Builder void __fastcall RemoveItem(TObject *Sender, ExplorerTreeLib_tlb::IGroup
*Group, ExplorerTreeLib_tlb::HITEM Item)
{
}
```

Delphi

```
procedure RemoveItem(ASender: TObject; Group : IGroup;Item : HITEM);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure RemoveItem(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveItemEvent);  
begin  
end;
```

Power...

```
begin event RemoveItem(oleobject Group,long Item)  
end event RemoveItem
```

VB.NET

```
Private Sub RemoveItem(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_RemoveItemEvent) Handles  
RemoveItem  
End Sub
```

VB6

```
Private Sub RemoveItem(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal Item  
As EXPLORERTREELibCtl.HITEM)  
End Sub
```

VBA

```
Private Sub RemoveItem(ByVal Group As Object,ByVal Item As Long)  
End Sub
```

VFP

```
LPARAMETERS Group,Item
```

Xbas...

```
PROCEDURE OnRemoveItem(oExplorerTree,Group,Item)  
RETURN
```

Syntax for RemoveItem event, **ICOM** version (others), on:

Java...

```
<SCRIPT EVENT="RemoveItem(Group,Item)" LANGUAGE="JScript">  
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript">  
Function RemoveItem(Group,Item)
```

```
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComRemoveItem Variant IIGroup HITEM IItem  
    Forward Send OnComRemoveItem IIGroup IItem  
End_Procedure
```

Visual
Objects

```
METHOD OCX_RemoveItem(Group,Item) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_RemoveItem(COM _Group,int _Item)  
{  
}
```

XBasic

```
function RemoveItem as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item as  
OLE::Exontrol.ExplorerTree.1::HITEM)  
end function
```

dBASE

```
function nativeObject_RemoveItem(Group,Item)  
return
```

event ScrollButtonClick (Group as Group, ScrollBar as ScrollBarEnum, ScrollPart as ScrollPartEnum)

Occurs when the user clicks a button in the scrollbar.

Type	Description
Group as Group	A Group object that indicates the group where the user clicked the scroll. If nothing, the user clicked the control's scroll bar, else it clicked the group's scroll bar
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that specifies the scrollbar being clicked.
ScrollPart as ScrollPartEnum	A ScrollPartEnum expression that indicates the part of the scroll being clicked.

Use the ScrollButtonClick event to notify your application that the user clicks a button in the control's scrollbar. The ScrollButtonClick event is fired when the user clicks and releases the mouse over an enabled part of the scroll bar. 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 [ScrollPartCaption](#) 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.

Syntax for ScrollButtonClick event, **/NET** version, on:

```
C# private void ScrollButtonClick(object sender,exontrol.EXPLORERTREELib.Group
Group,exontrol.EXPLORERTREELib.ScrollBarEnum
ScrollBar,exontrol.EXPLORERTREELib.ScrollPartEnum ScrollPart)
{
}
```

```
VB Private Sub ScrollButtonClick(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group,ByVal ScrollBar As
exontrol.EXPLORERTREELib.ScrollBarEnum,ByVal ScrollPart As
exontrol.EXPLORERTREELib.ScrollPartEnum) Handles ScrollButtonClick
End Sub
```

Syntax for ScrollButtonClick event, **/COM** version, on:

```
C# private void ScrollButtonClick(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_ScrollButtonClickEvent e)
{
```

```
}
```

C++

```
void OnScrollButtonClick(LPDISPATCH Group,long ScrollBar,long ScrollPart)
{
}
```

C++**Builder**

```
void __fastcall ScrollButtonClick(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::ScrollBarEnum
ScrollBar,Explorertreelib_tlb::ScrollPartEnum ScrollPart)
{
}
```

Delphi

```
procedure ScrollButtonClick(ASender: TObject; Group : IGroup;ScrollBar :
ScrollBarEnum;ScrollPart : ScrollPartEnum);
begin
end;
```

**Delphi 8
(.NET
only)**

```
procedure ScrollButtonClick(sender: System.Object; e:
AxEXPLORERTREELib._IExplorerTreeEvents_ScrollButtonClickEvent);
begin
end;
```

Powe...

```
begin event ScrollButtonClick(oleobject Group,long ScrollBar,long ScrollPart)
end event ScrollButtonClick
```

VB.NET

```
Private Sub ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXPLORERTREELib._IExplorerTreeEvents_ScrollButtonClickEvent) Handles
ScrollButtonClick
End Sub
```

VB6

```
Private Sub ScrollButtonClick(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal
ScrollBar As EXPLORERTREELibCtl.ScrollBarEnum,ByVal ScrollPart As
EXPLORERTREELibCtl.ScrollPartEnum)
End Sub
```

VBA

```
Private Sub ScrollButtonClick(ByVal Group As Object,ByVal ScrollBar As Long,ByVal
ScrollPart As Long)
End Sub
```

VFP

LPARAMETERS Group,ScrollBar,ScrollPart

Xbas...PROCEDURE OnScrollButtonClick(oExplorerTree,Group,ScrollBar,ScrollPart)
RETURNSyntax for ScrollButtonClick event, **/COM** version (others), on:**Java...**<SCRIPT EVENT="ScrollButtonClick(Group,ScrollBar,ScrollPart)"
LANGUAGE="JScript">
</SCRIPT>**VBSc...**<SCRIPT LANGUAGE="VBScript">
Function ScrollButtonClick(Group,ScrollBar,ScrollPart)
End Function
</SCRIPT>**Visual
Data...**Procedure OnComScrollButtonClick Variant IIGroup OLEScrollBarEnum IIScrollBar
OLEScrollPartEnum IIScrollPart
Forward Send OnComScrollButtonClick IIGroup IIScrollBar IIScrollPart
End_Procedure**Visual
Objects**METHOD OCX_ScrollButtonClick(Group,ScrollBar,ScrollPart) CLASS MainDialog
RETURN NIL**X++**void onEvent_ScrollButtonClick(COM _Group,int _ScrollBar,int _ScrollPart)
{
}**XBasic**function ScrollButtonClick as v (Group as
OLE::Exontrol.ExplorerTree.1::IGroup,ScrollBar as
OLE::Exontrol.ExplorerTree.1::ScrollBarEnum,ScrollPart as
OLE::Exontrol.ExplorerTree.1::ScrollPartEnum)
end function**dBASE**function nativeObject_ScrollButtonClick(Group,ScrollBar,ScrollPart)
return

event SelectGroup (Group as Group)

Occurs when a group is clicked.

Type	Description
Group as Group	A Group object being selected.

Use the SelectGroup event to notify your application that a new group is selected (clicked).

Syntax for SelectGroup event, **/NET** version, on:

```
C# private void SelectGroup(object sender,exontrol.EXPLORERTREELib.Group Group)
{
}
```

```
VB Private Sub SelectGroup(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group) Handles SelectGroup
End Sub
```

Syntax for SelectGroup event, **/COM** version, on:

```
C# private void SelectGroup(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_SelectGroupEvent e)
{
}
```

```
C++ void OnSelectGroup(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall SelectGroup(TObject *Sender,Explorertreelib_tlb::IGroup *Group)
{
}
```

```
Delphi procedure SelectGroup(ASender: TObject; Group : IGroup);
begin
end;
```


Delphi 8
(.NET
only)

```
procedure SelectGroup(sender: System.Object; e:  
AxEXPLOREERTREELib._IExplorerTreeEvents_SelectGroupEvent);  
begin  
end;
```

Powe...
begin event SelectGroup(oleobject Group)
end event SelectGroup

VB.NET
Private Sub SelectGroup(ByVal sender As System.Object, ByVal e As
AxEXPLOREERTREELib._IExplorerTreeEvents_SelectGroupEvent) Handles
SelectGroup
End Sub

VB6
Private Sub SelectGroup(ByVal Group As EXPLOREERTREELibCtl.IGroup)
End Sub

VBA
Private Sub SelectGroup(ByVal Group As Object)
End Sub

VFP
LPARAMETERS Group

Xbas...
PROCEDURE OnSelectGroup(oExplorerTree,Group)
RETURN

Syntax for SelectGroup event, **ICOM** version (others), on:

Java...
<SCRIPT EVENT="SelectGroup(Group)" LANGUAGE="JScript">
</SCRIPT>

VBSc...
<SCRIPT LANGUAGE="VBScript">
Function SelectGroup(Group)
End Function
</SCRIPT>

Visual
Data...

```
Procedure OnComSelectGroup Variant IIGroup  
    Forward Send OnComSelectGroup IIGroup  
End_Procedure
```

Visual
Objects

```
METHOD OCX_SelectGroup(Group) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_SelectGroup(COM _Group)  
{  
}
```

XBasic

```
function SelectGroup as v (Group as OLE::Exontrol.ExplorerTree.1::IIGroup)  
end function
```

dBASE

```
function nativeObject_SelectGroup(Group)  
return
```

The following sample focuses the group's list window when user selects the group:

```
Private Sub ExplorerTree1_SelectGroup(ByVal Group As EXPLOREERTREELibCtl.IIGroup)  
    Group.SetFocus  
End Sub
```

The following sample highlights the group being clicked, and restores the last selected group:

```
Dim gSelected As EXPLOREERTREELibCtl.Group  
  
Private Sub highGroup(ByVal g As EXPLOREERTREELibCtl.Group)  
    g.UserData = g.BackColor2  
    g.BackColor2 = vbBlue  
End Sub  
  
Private Sub unHighGroup(ByVal g As EXPLOREERTREELibCtl.Group)  
    g.BackColor2 = g.UserData  
End Sub
```

```
Private Sub ExplorerTree1_SelectGroup(ByVal Group As EXPLOREERTREELibCtl.IGroup)
    If Not gSelected.Index = Group.Index Then
        gSelected.Expanded = False
        unHighGroup gSelected
        Set gSelected = Group
        highGroup gSelected
    End If
End Sub
```

```
Private Sub Form_Load()
    Dim g As EXPLOREERTREELibCtl.Group
    ExplorerTree1.DelayScroll = 0
    ExplorerTree1.BeginUpdate
    For Each g In ExplorerTree1.Groups
        g.Expanded = False
        g.BackColor = ExplorerTree1.BackColorGroup
        g.BackColor2 = ExplorerTree1.BackColorGroup2
    Next
    Set gSelected = ExplorerTree1.Groups(0)
    gSelected.Expanded = True
    highGroup gSelected
    ExplorerTree1.EndUpdate
End Sub
```

event SelectionChanged (Group as Group)

Fired after a new item is selected.

Type	Description
Group as Group	A Group object where selection is changed.

The group supports multiple selection. When an item is selected or unselected the control fires the SelectionChanged event. Use the [SingleSel](#) property to specify if your control supports single or multiple selection. Use the [SelectCount](#) property to get the number of selected items within the group. Use the [SelectedItem](#) property to access the selected item by its index. Use the [SelectItem](#) property to select programmatically an item.

The following sample displays the list of selected items within the group:

```
Private Sub ExplorerTree1_SelectionChanged(ByVal Group As
EXPLORERTREELibCtl.IGroup)
    With Group.Items
        If (.SelectCount > 0) Then
            Dim i As Long
            For i = 0 To .SelectCount - 1
                Debug.Print .CellCaption(.SelectedItem(i), 0)
            Next
        End If
    End With
End Sub
```

Syntax for SelectionChanged event, **/NET** version, on:

```
C# private void SelectionChanged(object sender,exontrol.EXPLORERTREELib.Group
Group)
{
}
```

```
VB Private Sub SelectionChanged(ByVal sender As System.Object,ByVal Group As
exontrol.EXPLORERTREELib.Group) Handles SelectionChanged
End Sub
```

Syntax for SelectionChanged event, **/COM** version, on:

```
C# private void SelectionChanged(object sender,
```

```
AxEXPLORERTREELib._IExplorerTreeEvents_SelectionChangedEvent e)
{
}
```

```
C++ void OnSelectionChanged(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall SelectionChanged(TObject *Sender, Explorerlib_tlb::IGroup
*Group)
{
}
```

```
Delphi procedure SelectionChanged(ASender: TObject; Group : IGroup);
begin
end;
```

```
Delphi 8 (.NET only) procedure SelectionChanged(sender: System.Object; e:
AxEXPLORERTREELib._IExplorerTreeEvents_SelectionChangedEvent);
begin
end;
```

```
PowerBuilder begin event SelectionChanged(oleobject Group)
end event SelectionChanged
```

```
VB.NET Private Sub SelectionChanged(ByVal sender As System.Object, ByVal e As
AxEXPLORERTREELib._IExplorerTreeEvents_SelectionChangedEvent) Handles
SelectionChanged
End Sub
```

```
VB6 Private Sub SelectionChanged(ByVal Group As EXPLORERTREELibCtl.IGroup)
End Sub
```

```
VBA Private Sub SelectionChanged(ByVal Group As Object)
End Sub
```

```
VFP LPARAMETERS Group
```

```
Xbas... PROCEDURE OnSelectionChanged(oExplorerTree,Group)
RETURN
```

Syntax for SelectionChanged event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="SelectionChanged(Group)" LANGUAGE="JScript">
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">
Function SelectionChanged(Group)
End Function
</SCRIPT>
```

```
Visual Data... Procedure OnComSelectionChanged Variant IIGroup
Forward Send OnComSelectionChanged IIGroup
End_Procedure
```

```
Visual Objects METHOD OCX_SelectionChanged(Group) CLASS MainDialog
RETURN NIL
```

```
X++ void onEvent_SelectionChanged(COM _Group)
{
}
```

```
XBasic function SelectionChanged as v (Group as OLE::Exontrol.ExplorerTree.1::IIGroup)
end function
```

```
dBASE function nativeObject_SelectionChanged(Group)
return
```

event SelectShortcut (OldShortcut as Variant, NewShortcut as Variant)

Fired when the user selects a new shortcut.

Type	Description
OldShortcut as Variant	A String expression that indicates the caption of the shortcut being unselected.
NewShortcut as Variant	A String expression that indicates the caption of the shortcut being unselected.

The SelectShortcut event notifies your application when the user selects a shortcut. The SelectShortcut event is fired if the user clicks a shortcut in the shortcut bar, or if the code calls the [SelectShortcut](#) property. The [ShowShortcutBar](#) property shows or hides the control's shortcut bar. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut.

Syntax for SelectShortcut event, **/NET** version, on:

```
C# private void SelectShortcut(object sender,object OldShortcut,object  
NewShortcut)  
{  
}
```

```
VB Private Sub SelectShortcut(ByVal sender As System.Object,ByVal OldShortcut As  
Object,ByVal NewShortcut As Object) Handles SelectShortcut  
End Sub
```

Syntax for SelectShortcut event, **/COM** version, on:

```
C# private void SelectShortcut(object sender,  
AxEXPLORERTREELib._IExplorerTreeEvents_SelectShortcutEvent e)  
{  
}
```

```
C++ void OnSelectShortcut(VARIANT OldShortcut,VARIANT NewShortcut)  
{  
}
```

```
C++  
Builder void __fastcall SelectShortcut(TObject *Sender,Variant OldShortcut,Variant  
NewShortcut)
```

```
{  
}
```

```
Delphi  
procedure SelectShortcut(ASender: TObject; OldShortcut :  
OleVariant;NewShortcut : OleVariant);  
begin  
end;
```

```
Delphi 8  
(.NET  
only)  
procedure SelectShortcut(sender: System.Object; e:  
AxEXPLORERTREELib._IExplorerTreeEvents_SelectShortcutEvent);  
begin  
end;
```

```
Powe...  
begin event SelectShortcut(any OldShortcut,any NewShortcut)  
end event SelectShortcut
```

```
VB.NET  
Private Sub SelectShortcut(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib._IExplorerTreeEvents_SelectShortcutEvent) Handles  
SelectShortcut  
End Sub
```

```
VB6  
Private Sub SelectShortcut(ByVal OldShortcut As Variant,ByVal NewShortcut As  
Variant)  
End Sub
```

```
VBA  
Private Sub SelectShortcut(ByVal OldShortcut As Variant,ByVal NewShortcut As  
Variant)  
End Sub
```

```
VFP  
LPARAMETERS OldShortcut,NewShortcut
```

```
Xbas...  
PROCEDURE OnSelectShortcut(oExplorerTree,OldShortcut,NewShortcut)  
RETURN
```

Syntax for SelectShortcut event, **ICOM** version (others), on:

```
Java...  
<SCRIPT EVENT="SelectShortcut(OldShortcut,NewShortcut)"
```



```
LANGUAGE="JScript" >  
</SCRIPT>
```

VBS...

```
<SCRIPT LANGUAGE="VBScript" >  
Function SelectShortcut(OldShortcut,NewShortcut)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComSelectShortcut Variant IIOldShortcut Variant IINewShortcut  
Forward Send OnComSelectShortcut IIOldShortcut IINewShortcut  
End_Procedure
```

Visual
Objects

```
METHOD OCX_SelectShortcut(OldShortcut,NewShortcut) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_SelectShortcut(COMVariant _OldShortcut,COMVariant  
_NewShortcut)  
{  
}
```

XBasic

```
function SelectShortcut as v (OldShortcut as A,NewShortcut as A)  
end function
```

dBASE

```
function nativeObject_SelectShortcut(OldShortcut,NewShortcut)  
return
```

event ToolTip (Group as Group, Item as HITEM, ColIndex as Long, ByRef Visible as Boolean, ByRef X as Long, ByRef Y as Long, CX as Long, CY as Long)

Fired when the control prepares the object's tooltip.

Type	Description
Group as Group	A Group object where the tooltip is about to appear
Item as HITEM	A long expression that indicates the item's handle or 0 if the cursor is not over the cell.
ColIndex as Long	A long expression that indicates the column's index.
Visible as Boolean	(By Reference) A boolean expression that indicates whether the object's tooltip is visible.
X as Long	(By Reference) A long expression that indicates the left location of the tooltip window. The x values is always expressed in screen coordinates.
Y as Long	(By Reference) A long expression that indicates the top location of the tooltip window. The y values is always expressed in screen coordinates.
CX as Long	A long expression that indicates the width of the tooltip window.
CY as Long	A long expression that indicates the height of the tooltip window.

The ToolTip event notifies your application that the control prepares the tooltip for a cell or column. Use the ToolTip event to change the default position of the tooltip window. Use the [CellToolTip](#) property to specify the cell's tooltip. Use the [Tooltip](#) property to assign a tooltip to a column. Use the [ToolTipWidth](#) property to specify the width of the tooltip window. The [ToolTipDelay](#) property specifies the time in ms that passes before the ToolTip appears. The [ToolTipPopDelay](#) specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.

Syntax for ToolTip event, **/.NET** version, on:

C#

```
private void ToolTip(object sender,exontrol.EXPLORERTREELib.Group Group,int  
Item,int ColIndex,ref bool Visible,ref int X,ref int Y,int CX,int CY)  
{  
}
```

VB

```
Private Sub ToolTip(ByVal sender As System.Object,ByVal Group As
```

```
exontrol.EXPLORERTREELib.Group,ByVal Item As Integer,ByVal ColIndex As Integer,ByRef Visible As Boolean,ByRef X As Integer,ByRef Y As Integer,ByVal CX As Integer,ByVal CY As Integer) Handles ToolTip
End Sub
```

Syntax for ToolTip event, **/COM** version, on:

```
C# private void ToolTip(object sender,
AxEXPLORERTREELib._IExplorerTreeEvents_ToolTipEvent e)
{
}
```

```
C++ void OnToolTip(LPDISPATCH Group,long Item,long ColIndex,BOOL FAR*
Visible,long FAR* X,long FAR* Y,long CX,long CY)
{
}
```

```
C++ Builder void __fastcall ToolTip(TObject *Sender,Explorertreelib_tlb::IGroup
*Group,Explorertreelib_tlb::HITEM Item,long ColIndex,VARIANT_BOOL *
Visible,long * X,long * Y,long CX,long CY)
{
}
```

```
Delphi procedure ToolTip(ASender: TObject; Group : IGroup;Item : HITEM;ColIndex :
Integer;var Visible : WordBool;var X : Integer;var Y : Integer;CX : Integer;CY :
Integer);
begin
end;
```

```
Delphi 8 (.NET only) procedure ToolTip(sender: System.Object; e:
AxEXPLORERTREELib._IExplorerTreeEvents_ToolTipEvent);
begin
end;
```

```
Power... begin event ToolTip(oleobject Group,long Item,long ColIndex,boolean
Visible,long X,long Y,long CX,long CY)
end event ToolTip
```

```
VB.NET Private Sub ToolTip(ByVal sender As System.Object, ByVal e As  
AxEXPLORERTREELib.IExplorerTreeEvents.ToolTipEvent) Handles ToolTip  
End Sub
```

```
VB6 Private Sub ToolTip(ByVal Group As EXPLORERTREELibCtl.IGroup,ByVal Item As  
EXPLORERTREELibCtl.HITEM,ByVal ColIndex As Long,Visible As Boolean,X As  
Long,Y As Long,ByVal CX As Long,ByVal CY As Long)  
End Sub
```

```
VBA Private Sub ToolTip(ByVal Group As Object,ByVal Item As Long,ByVal ColIndex As  
Long,Visible As Boolean,X As Long,Y As Long,ByVal CX As Long,ByVal CY As Long)  
End Sub
```

```
VFP LPARAMETERS Group,Item,ColIndex,Visible,X,Y,CX,CY
```

```
Xbas... PROCEDURE OnToolTip(oExplorerTree,Group,Item,ColIndex,Visible,X,Y,CX,CY)  
RETURN
```

Syntax for ToolTip event, **ICOM** version (others), on:

```
Java... <SCRIPT EVENT="ToolTip(Group,Item,ColIndex,Visible,X,Y,CX,CY)"  
LANGUAGE="JScript">  
</SCRIPT>
```

```
VBS... <SCRIPT LANGUAGE="VBScript">  
Function ToolTip(Group,Item,ColIndex,Visible,X,Y,CX,CY)  
End Function  
</SCRIPT>
```

```
Visual Data... Procedure OnComToolTip Variant IIGroup HITEM IItem Integer IIColIndex Boolean  
IIVisible Integer IIX Integer IYY Integer IICX Integer IICY  
Forward Send OnComToolTip IIGroup IItem IIColIndex IIVisible IIX IYY IICX IICY  
End_Procedure
```

```
Visual Objects METHOD OCX_ToolTip(Group,Item,ColIndex,Visible,X,Y,CX,CY) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_ToolTip(COM _Group,int _Item,int _CollIndex,COMVariant /*bool*/  
_Visible,COMVariant /*long*/ _X,COMVariant /*long*/ _Y,int _CX,int _CY)  
{  
}
```

XBasic

```
function ToolTip as v (Group as OLE::Exontrol.ExplorerTree.1::IGroup,Item as  
OLE::Exontrol.ExplorerTree.1::HITEM,CollIndex as N,Visible as L,X as N,Y as N,CX as  
N,CY as N)  
end function
```

dBASE

```
function nativeObject_ToolTip(Group,Item,CollIndex,Visible,X,Y,CX,CY)  
return
```