ExOrgChart

The ExOrgChart component permits the totally automatic generation of organigrams. The ExOrgChart component lets the user changes its visual appearance using **skins**, each one providing an additional visual experience that enhances viewing pleasure. Skins are relatively easy to build and put on any part of the control.

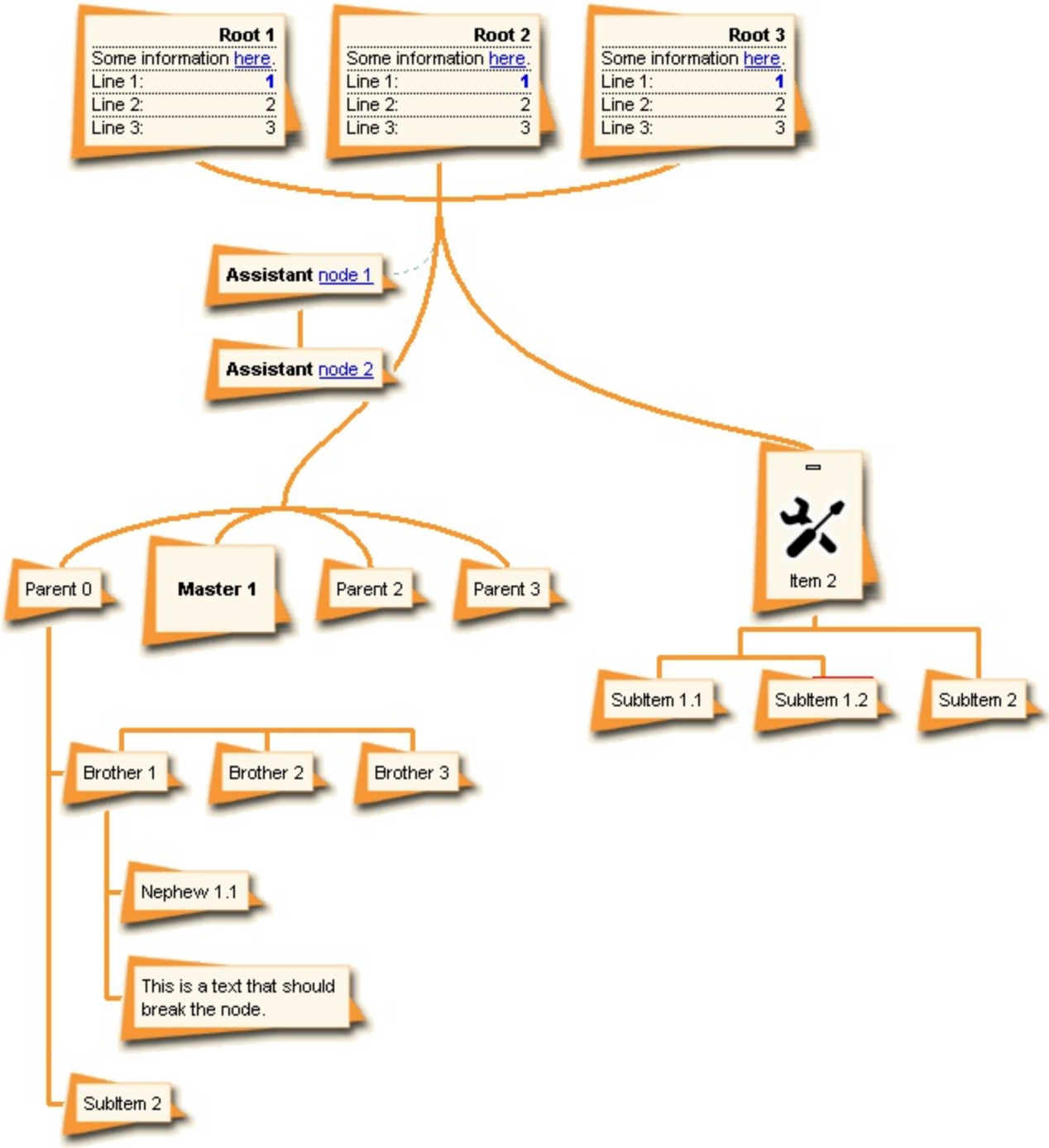
Features include:

- **WYSWYG Template/Layout** Editor support
- **Skinnable Interface** support (ability to apply a skin to the any background part)
- Print and **Print Preview** Support, Fit-To-Pages Wide By Tall, ...
- Ability to save/load the control's data to/from **XML** files
- Top to Bottom (TTB), Left to Right (LTR) layouts support
- Ability to specify **multiple parents** or multiple roots
- Ability to save the control's content to x-script template.
- Ability to arrange the child nodes **horizontally**, **vertically** or as a **tree**
- Ability to show frame(s) around any node or group of nodes
- Ability to expand or collapse the nodes
- Ability to copy the control's content to the clipboard
- Ability to load the pictures, icons using the BASE64 encoded strings
- Ability to put any HTML caption on any link (including pictures, icons, anchors and so on)
- Support for **assistant** nodes
- Ability to **zoom** the chart
- Multiple lines **ToolTip** support
- Background picture support
- Mouse wheel support
- Ability to explore only a branch of the organigram
- Ability to display single or multiple layers of the organigram
- Ability to scroll the organigram using the mouse or keyboard
- Unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the event's background.
- Any node supports built-in HTML format
- Ability to insert **hyperlinks** anywhere in the node's caption
- The node supports multiple lines with different alignments
- Ability to assign icons, images, pictures to the node aligned to any side of it.
- Ability to link a node with any other nodes.
- Ability to show directions of the links, HTML captions and so on
- Ability to show smoothly the lines and curves in the control, using antialiasing rendering
- BMP, EMF, EXIF, GIF, ICON, JPEG, PNG, TIFF or WMF formats for pictures, icons

or images.

- AutoSize feature or fixed width, fixed height for any node supported
- The links and borders for nodes are customizable
- and more

The ANSI and UNICODE version are available.



How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

The AlignmentEnum type aligns the text inside the object.

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

constants AllowKeysEnum

The AllowKeysEnum type specifies the keys to be combined in order to start an UI operation. For instance, the AllowEdit property of AllowKeysEnum type indicates the keys combination to let user edit the node's caption at runtime. By default, this property is set on exLeftClick + exDblLick, which means the user is able to edit the node's caption by double clicking the left mouse button. If this property is set on exRightClick + exCRTLKey the user should press the CTRL key while right clicking the control to start editing the node. If the exDblClick flag is included, the user requires to do a double click instead single click to perform the operation. The exDisallow flag indicates that the operation is not allowed.

The AllowKeysEnum type supports the following values:

Name	Value	Description
exDisallow	0	The operation is not allowed.
exLeftClick	1	The operation starts once the user clicks the left mouse button.
exRightClick	2	The operation starts if the user clicks the right mouse button.
exMiddleClick	3	The operation starts if the user clicks the middle mouse button.
exSHIFTKey	8	The operation may start only if the user presses the SHIFT key.
exCTRLKey	16	The operation may start only if the user presses the CTRL key.
exALTKey	32	The operation may start only if the user presses the ALT key.
exDblClick	64	The operation starts only if the user double clicks, instead single click.

constants AppearanceEnum

Specifies the control's appearance. Use the Appearance property to specify the control's appearance.

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

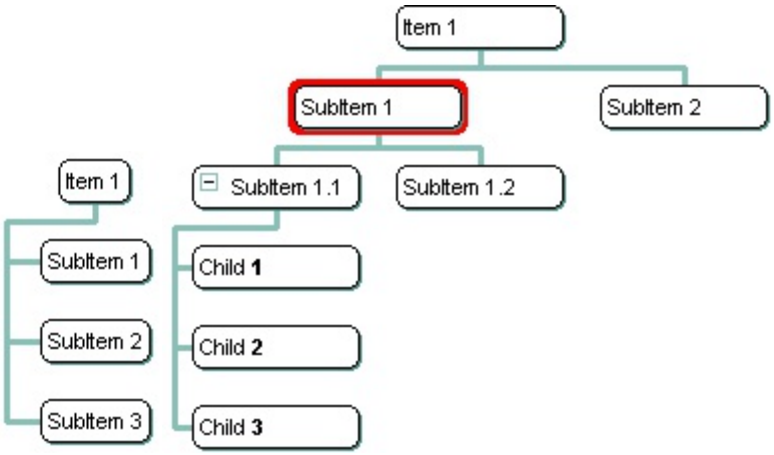
constants ArrangeSiblingEnum

The ArrangeSiblingEnum expression specifies how the first child node and its sibling nodes are arranged. Use the ArrangeSiblingNodesAs property to specify how the child nodes of the node are arranged.

Name	Value	Description
exDefault	0	The first child node and its sibling nodes are arranged vertically.
exHorizontally	1	The first child node and its sibling nodes are arranged horizontally.
exTree	2	Aligns the child nodes as a tree, so each node is indented relative to its parent.

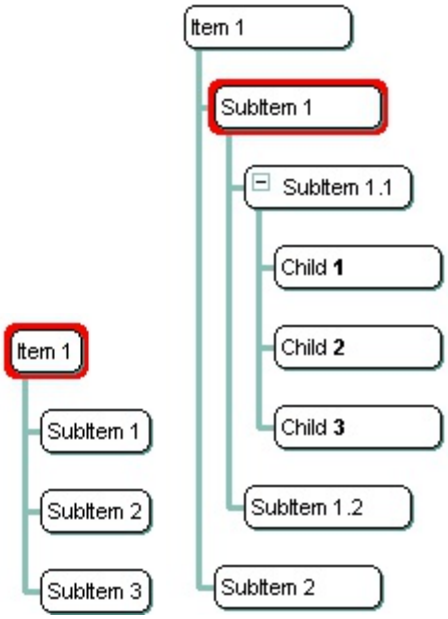
For instance, let's say that we have a parent item named Item1, and three child nodes, SubItem 1, SubItem 2 and SubItem 3.

The **exDefault** arrangement looks like follows:



The **exHorizontally** arrangement looks like follows:

The **exTree** arrangement looks like follows:


Item 1
SubItem 1
SubItem 2
SubItem 3
Item 1
SubItem 1
SubItem 1.1
Child 1
Child 2
Child 3
SubItem 1.2
SubItem 2

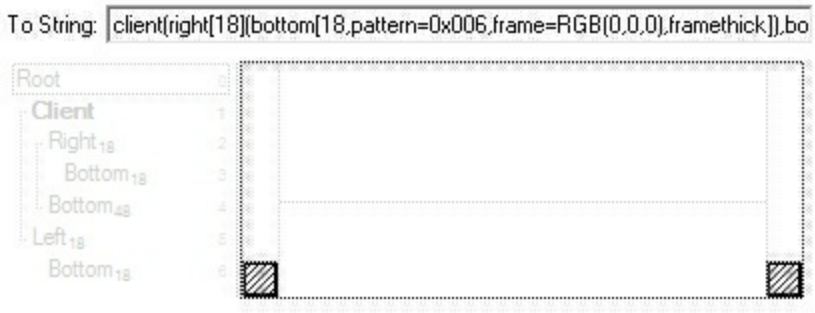
constants AspectRatioEnum

The AspectRatioEnum expression specifies the aspect ratio for the picture being displayed in the node. The PictureAspectRatio property specifies the aspect ratio for the node's picture. The Picture property assigns a picture to a node. The PictureWidth and PictureHeight properties controls the size of displayed picture.

Name	Value	Description
exAspectRatioNone	0	No aspect ratio is applied to the node's picture.
exAspectRatioWidth	1	The picture's height is computed based on the original size of the picture while specifying the picture's width.
exAspectRatioHeight	2	The picture's width is computed based on the original size of the picture while specifying the picture's height.

constants BackgroundExtPropertyEnum

The BackgroundExtPropertyEnum type specifies the UI properties of the part of the EBN you can access/change at runtime. The BackgroundExt property specifies the EBN String format to be displayed on the node's background. The BackgroundExtValue property access the value of the giving property for specified part of the EBN. The BackgroundExtPropertyEnum type supports the following values:

Name	Value	Description
exToStringExt	0	Specifies the part's ToString representation. The BackgroundExt property specifies the EBN String format to be displayed on the object's background. *The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field.* *Sample:* *"client(right[18](bottom[18,pattern=6,frame=0,framethick]),bottom[4(bottom[18,pattern=6,frame=0,framethick])"* *generates the following layout:*  *where it is applied to an object it looks as follows:*  *(String expression, read-only).*

exBackColorExt	1	Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000.* *(Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*
		Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect.* *Based on the exAnchorExt value the exClientExt is:* • *0 (**none**, the object is not anchored to any side), the format of the exClientExt is **"left,top,width,height"** (as string) where (left,top) margin indicates the position where the part starts, and the (width,height) pair specifies its size. The left, top, width or height could be any expression (+,-,/ or *) that can include numbers associated with pixels or percents. For instance: "25%,25%,50%,50%" indicates the middle of the parent object, and so when the parent is resized the client is resized accordingly. The "50%-8,50%-8,16,16" value specifies that the size of the object is always 16x16 pixels and positioned on the center of the parent object.* • *1 (**left**, the object is anchored to left side of the parent), the format of the exClientExt is **width** (string or numeric) where width indicates the width of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates*

exClientExt	2	*the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *2 (**right**, the object is anchored to right side of the parent object), the format of the exClientExt is **width** (string or numeric) where width indicates the width of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *3 (**client**, the object takes the full available area of the parent), the exClientExt has no effect.* • *4 (**top**, the object is anchored to the top side of the parent object), the format of the exClientExt is **height** (string or numeric) where height indicates the height of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *5 (**bottom**, the object is anchored to bottom side of the parent object), the format of the exClientExt is **height** (string or numeric) where height indicates the height of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* *Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent.*

		(String/Numeric expression)
exAnchorExt	3	Specifies the object's alignment relative to its parent. *The valid values for exAnchorExt are:* • *0 **(none)**, the object is not anchored to any side,* • *1 **(left)**, the object is anchored to left side of the parent,* • *2 **(right)**, the object is anchored to right side of the parent object,* • *3 **(client)**, the object takes the full available area of the parent,* • *4 **(top)**, the object is anchored to the top side of the parent object,* • *5 **(bottom)**, the object is anchored to bottom side of the parent object* *(Numeric expression)*
		Specifies the HTML text to be displayed on the object. *The exTextExt supports the following built-in HTML tags:* • ** ... ** displays the text in **bold** • ** ... ** displays the text in *italics* • **<u> ... </u>** underlines the text • **<s> ... </s>** ~~Strike~~-through text • **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The

FormatAnchor property customizes the visual effect for anchor elements.

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=gA8ABmABnABjABvABshIAOQAEAA " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAA 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=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break

exTextExt 4

- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the

Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the

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

gradient-center

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

outlined

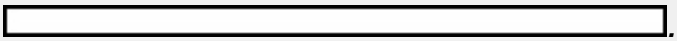
- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2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

shadow

or "*<**sha** 404040;5;0>*

		*<fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:  *(String expression)*
exTextExtWordWrap	5	Specifies that the object is wrapping the text. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *(Boolean expression)*
exTextExtAlignment	6	Indicates the alignment of the text on the object. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *The valid values for exTextExtAlignment are:* • *0, (hexa 0x00, **Top-Left**), Text is vertically aligned at the top, and horizontally aligned on the left.* • *1, (hexa 0x01, **Top-Center**), Text is vertically aligned at the top, and horizontally aligned at the center.* • *2, (hexa 0x02, **Top-Right**), Text is vertically aligned at the top, and horizontally aligned on the right.* • *16, (hexa 0x10, **Middle-Left**), Text is vertically aligned in the middle, and horizontally aligned on the left.* • *17, (hexa 0x11, **Middle-Center**), Text is vertically aligned in the middle, and horizontally aligned at the center.* • *18, (hexa 0x12, **Middle-Right**), Text is vertically aligned in the middle, and horizontally aligned on the right.*

		• *32, (hexa 0x20, **Bottom-Left**), Text is vertically aligned at the bottom, and horizontally aligned on the left.* • *33, (hexa 0x21, **Bottom-Center**), Text is vertically aligned at the bottom, and horizontally aligned at the center.* • *34, (hexa 0x22, **Bottom-Right**), Text is vertically aligned at the bottom, and horizontally aligned on the right.* *(Numeric expression)*
exPatternExt	7	Indicates the pattern to be shown on the object. The exPatternColorExt specifies the color to show the pattern. *The valid values for exPatternExt are:* • *0, (hexa 0x000, **Empty**), The pattern is not visible* • *1, (hexa 0x001, **Solid**),* • *2, (hexa 0x002, **Dot**),* • *3, (hexa 0x003, **Shadow**),* • *4, (hexa 0x004, **NDot**),* • *5, (hexa 0x005, **FDiagonal**),* • *6, (hexa 0x006, **BDiagonal**),* • *7, (hexa 0x007, **DiagCross**),* • *8, (hexa 0x008, **Vertical**),* • *9, (hexa 0x009, **Horizontal**),* • *10, (hexa 0x00A, **Cross**),* • *11, (hexa 0x00B, **Brick**),*

		• *12, (hexa 0x00C, **Yard**),*  • *256, (hexa 0x100, **Frame**),*  *. The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags.* • *768, (hexa 0x300, **FrameThick**),*  *. The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags.* *(Numeric expression)*
exPatternColorExt	8	Indicates the color to show the pattern on the object. The exPatternColorExt property has effect only if the exPatternExt property is not 0 (empty). The exFrameColorExt specifies the color to show the frame (the exPatternExt property includes the exFrame or exFrameThick flag) *(Color expression)*
exFrameColorExt	9	Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. *(Color expression)*
exFrameThickExt	10	Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property. *(Boolean expression)*
exUserDataExt	11	Specifies an extra-data associated with the object. (*Variant expression)*

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
exNodeFrame	0	Specifies the node's frame. The DrawRoundNode property specifies a value that indicates whether the node has borders with round corners. The ShadowNode property determines whether the control displays a shadow for nodes. The DrawRoundNode property and ShadowNode property has effect only if no skin is applied to a node.
exEditNodeBackColor	1	Specifies the edit's background. The AllowEdit property specifies the combination of keys that allows the user to edit a node. The LayoutStartChanging(exEditNode) event notifies your application once the user starts editing the node's caption.
		Specifies the edit's foreground. The AllowEdit property specifies the combination of keys that

exEditNodeForeColor	2	allows the user to edit a node. The LayoutStartChanging(exEditNode) event notifies your application once the user starts editing the node's caption. The Caption property indicates the caption of the node being edited.
exToolTipAppearance	64	Indicates the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTip property to specify the cell's tooltip. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ShowToolTip method to display a custom tooltip.
exToolTipBackColor	65	exToolTipBackColor. Specifies the tooltip's background color.
exToolTipForeColor	66	exToolTipForeColor. Specifies the tooltip's foreground color.
exContextMenuAppearance	99	exContextMenuAppearance. Specifies the visual appearance of the control's context menu.
exContextMenuBackColor	100	exContextMenuBackColor. Specifies the solid background color for the control's context menu.
exContextMenuForeColor	101	exContextMenuForeColor. Specifies the text foreground color for the control's context menu.
exContextMenuSelBackColor	102	exContextMenuSelBackColor. Specifies the solid/EBN selection's background color in the control's context menu.
exContextMenuSelBorderColor	103	exContextMenuSelBorderColor. Specifies the solid color to show the selection in the control's context menu.
exContextMenuSelForeColor	104	exContextMenuSelForeColor. Specifies the selection's text foreground color in the control's context menu.
exSizeGrip	3	Specifies the visual appearance for control's size grip.
exVSUp	256	The up button in normal state.
exVSUpP	257	The up button when it is pressed.

exVSUpD	258	The up button when it is disabled.
exVSUpH	259	The up button when the cursor hovers it.
exVSThumb	260	The thumb part (exThumbPart) in normal state.
exVSThumbP	261	The thumb part (exThumbPart) when it is pressed.
exVSThumbD	262	The thumb part (exThumbPart) when it is disabled.
exVSThumbH	263	The thumb part (exThumbPart) when cursor hovers it.
exVSDown	264	The down button in normal state.
exVSDownP	265	The down button when it is pressed.
exVSDownD	266	The down button when it is disabled.
exVSDownH	267	The down button when the cursor hovers it.
exVSLower	268	The lower part (exLowerBackPart) in normal state.
exVSLowerP	269	The lower part (exLowerBackPart) when it is pressed.
exVSLowerD	270	The lower part (exLowerBackPart) when it is disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.
exVSUpper	272	The upper part (exUpperBackPart) in normal state.
exVSUpperP	273	The upper part (exUpperBackPart) when it is pressed.
exVSUpperD	274	The upper part (exUpperBackPart) when it is disabled.
exVSUpperH	275	The upper part (exUpperBackPart) when the cursor hovers it.
exVSBack	276	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exVSBackP	277	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exVSBackD	278	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exVSBackH	279	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.

exHSLeft	384	The left button in normal state.
exHSLeftP	385	The left button when it is pressed.
exHSLeftD	386	The left button when it is disabled.
exHSLeftH	387	The left button when the cursor hovers it.
exHSThumb	388	The thumb part (exThumbPart) in normal state.
exHSThumbP	389	The thumb part (exThumbPart) when it is pressed.
exHSThumbD	390	The thumb part (exThumbPart) when it is disabled.
exHSThumbH	391	The thumb part (exThumbPart) when the cursor hovers it.
exHSRight	392	The right button in normal state.
exHSRightP	393	The right button when it is pressed.
exHSRightD	394	The right button when it is disabled.
exHSRightH	395	The right button when the cursor hovers it.
exHSLower	396	The lower part (exLowerBackPart) in normal state.
exHSLowerP	397	The lower part (exLowerBackPart) when it is pressed.
exHSLowerD	398	The lower part (exLowerBackPart) when it is disabled.
exHSLowerH	399	The lower part (exLowerBackPart) when the cursor hovers it.
exHSUpper	400	The upper part (exUpperBackPart) in normal state.
exHSUpperP	401	The upper part (exUpperBackPart) when it is pressed.
exHSUpperD	402	The upper part (exUpperBackPart) when it is disabled.
exHSUpperH	403	The upper part (exUpperBackPart) when the cursor hovers it.
exHSBack	404	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exHSBackP	405	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exHSBackD	406	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.

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

constants CaptionSingleLineEnum

The CaptionSingleLineEnum type defines whether the element's caption is displayed on a single or multiple lines. The CaptionSingleLine property retrieves or sets a value indicating whether the element's (extra/)caption is displayed using one line, or multiple lines. The CaptionSingleLineEnum type supports the following values:

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

constants ChartLayoutEnum

The ChartLayoutEnum type specifies the way the nodes are arranged in the chart. The Layout property of the control specifies whether the nodes are arranged from the Top to Bottom (TTB) or from Left to Right (LTR).

Name	Value	Description
exLayoutTTB	0	The chart displays the nodes from top to bottom (TTB layout). (by default)
exLayoutLTR	1	The chart displays the nodes from left to right (LTR layout).

constants ClientAreaEnum

The ClientAreaEnum type specifies the area inside the control. Use the Cursor property to change the mouse pointer when cursor hovers the control

Name	Value	Description
exChartArea	0	The cursor is on the control's chart area.
exNodeArea	1	The cursor hovers a node in the chart.
exDragChart	2	The user drags the chart.
exExpandButtonArea	3	The cursor hovers an expanding/collapsing button.

constants EditableNodeEnum

The EditableNodeEnum type specifies whether the node is editable at runtime. Use the Editable property to specify whether a specified node can be edited at runtime. The AllowEdit property specifies the combination of keys that allows the user to edit a node. The LayoutStartChanging(exEditNode) event notifies your application once the user starts editing the node's caption. The LayoutEndChanging(exEditNode) event notifies your application once of the edit operation ends.

Name	Value	Description
exNoEditable	0	The operation is not allowed.
exEditable	1	The inline editing is allowed and it ends once the user double/clicks the node (AllowEdit property)

constants ExpandButtonEnum

Specifies the appearance for +/- buttons being displayed if the HasButtons property is not null.

Name	Value	Description
exNoButtons	0	The control displays no expand buttons.
exPlus	-1	A plus sign is displayed for collapsed nodes, and a minus sign for expanded nodes.(⊞ ⊟)
exArrow	1	The control uses icons to display the expand buttons.(▸ ▾)
exCircle	2	The control uses icons to display the expand buttons. (⊕ ○)
exWPlus	3	The control uses icons to display the expand buttons. (✚ ▬)
exCustom	4	The HasButtonsCustom property specifies the index of icons being used for +/- signs on parent nodes.

constants LayoutChangingEnum

The LayoutStartChanging event notifies your application once the user starts any of these operations. The LayoutEndChanging event notifies your application once of the following operation ends. The LayoutChangingEnum type supports the following values:

Name	Value	Description
exEditNode	0	Notifies your application that a node is being inline edited at runtime. The Caption property indicates the caption of the node being edited. Use the NodeFromPoint property to get the node from the current position. The AllowEdit property indicates the keys combination so the user start editing the node's caption at runtime.
exResizeChart	1	Occurs once the chart is magnified or shrank, at runtime. Use the ZoomWidthMode property to specify whether the ZoomWidth property is updated when the control is resized. Use the ZoomHeightMode property to specify whether the ZoomHeight property is updated when the control is resized.
exMoveNode	2	Occurs once user moves a node at runtime. The Parent property specifies the parent node. Use the NodeFromPoint property to get the node from the current position. The DragOutsideDef property customizes the speed to scroll the control's content while user moves nodes by drag and drop then outside of the control's client area.

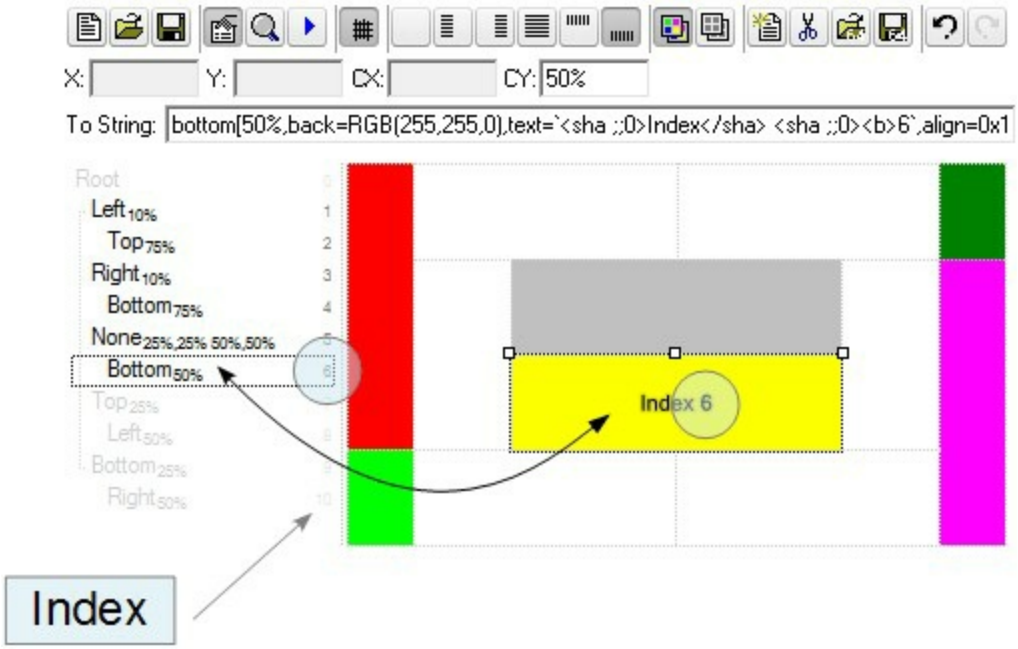
constants ImageAlignmentEnum

Specifies the alignment of the image.

Name	Value	Description
exImageLeft	0	The images is aligned to the left side of the object.
exImageRight	1	The images is aligned to the right side of the object.
exImageTop	2	The images is aligned to the top side of the object.
exImageBottom	3	The images is aligned to the bottom side of the object.

constants IndexExtEnum

The IndexExtEnum type specifies the index of the part of the EBN object to be accessed. The Index parameter of the BackgroundExtValue property indicates the index of the part of the EBN object to be changed or accessed. *The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field. The list of objects that compose the EBN are displayed on the left side of the Builder tool, and the Index of the part is displayed on each item aligned to the right as shown in the following screen shot:*

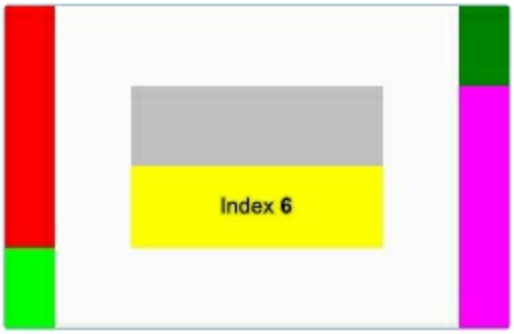


In this sample, there are 11 objects that composes the EBN, so the Index property goes from 0 which indicates the root, and 10, which is the last item in the list

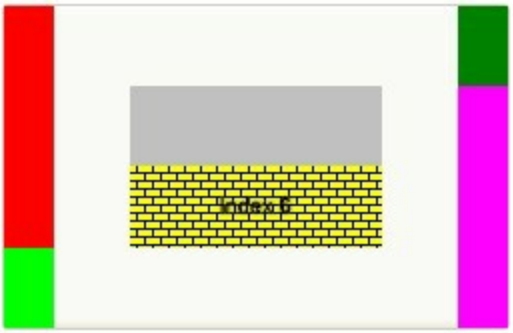
So, let's apply this format to an object, to change the exPatternExt property for the object with the Index 6:

Before calling the BackgroundExt property:

After calling the BackgroundExt property:



and now, let's change the exPatternExt property of the object with the Index 6 to 11 (Yard), so finally we got:



The IndexExtEnum type supports the following values:

Name	Value	Description
exIndexExtRoot	0	Specifies the part of the object with the index 0 (root).
exIndexExt1	1	Specifies the part of the object with the index 1.
exIndexExt2	2	Specifies the part of the object with the index 2.
exIndexExt3	3	Specifies the part of the object with the index 3.
exIndexExt4	4	Specifies the part of the object with the index 4.
exIndexExt5	5	Specifies the part of the object with the index 5.
exIndexExt6	6	Specifies the part of the object with the index 6.
exIndexExt7	7	Specifies the part of the object with the index 7.

constants PaddingEdgeEnum

The PaddingEdgeEnum type defines the margins of the object. The DefaultNodePadding property defines the padding for all nodes. Use the Padding property of the Node to define the padding for specified node. The PaddingEdgeEnum type supports the following values:

Name	Value	Description
exPaddingAll	-1	Indicates all margins of the object.
exPaddingLeft	0	Indicates the left margin of the object.
exPaddingTop	1	Indicates the top margin of the object.
exPaddingRight	2	Indicates the right margin of the object.
exPaddingBottom	3	Indicates the bottom margin of the object.

constants PatternEnum

The PatternEnum type specifies the type of patterns that the element can fill with. The Type property indicates the pattern to fill the element. The Color property indicates the color to fill the element's pattern, while the FrameColor property indicates the color to show the element's border/frame if the Type property includes the exPatternFrame flag.

The PatternEnum type supports the following values:

Name	Value	Description
exPatternEmpty	0	exPatternEmpty
exPatternSolid	1	exPatternSolid
exPatternDot	2	exPatternDot
exPatternShadow	3	exPatternShadow
exPatternNDot	4	exPatternNDot
exPatternFDiagonal	5	exPatternFDiagonal
exPatternBDiagonal	6	exPatternBDiagonal
exPatternDiagCross	7	exPatternDiagCross
exPatternVertical	8	exPatternVertical
exPatternHorizontal	9	exPatternHorizontal
exPatternCross	10	exPatternCross
exPatternBrick	11	exPatternBrick
exPatternYard	12	exPatternYard
exPatternFrame	256	exPatternFrame
exPatternFrameThick	768	exPatternFrameThick

constants PenTypeEnum

Specifies the type of the pen. Use the PenLink property to specify the type of the pen used to paint the links between nodes. Use the PenWidthLink property to specify the width of the pen.

Name	Value	Description
exPenSolid	0	The pen is solid.
exPenDash	1	The pen is dashed. This style is valid only when the pen width is one or less in device units.
exPenDot	2	The pen is dotted. This style is valid only when the pen width is one or less in device units.
exPenDashDot	3	The pen has alternating dashes and dots. This style is valid only when the pen width is one or less in device units.
exPenDashDotDot	4	The pen has alternating dashes and double dots. This style is valid only when the pen width is one or less in device units.

constants PictureDisplayEnum

Aligns the picture in the control. Use the Picture property to assign a picture on the control's background. Use the PictureDisplay property to specify the way how the picture is arranged on the control's backgroun.

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

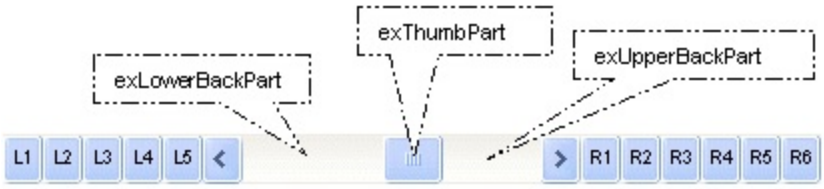
constants ScrollBarEnum

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

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

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 forth additional button, in the left or top area. By default, this button is hidden.
exLeftB5Part	2048	(L5) The fifth additional button, in the left or top area. By default, this button is hidden.
exLeftBPart	1024	(<) The left or top button. By default, this button is visible.
exLowerBackPart	512	The area between the left/top button and the thumb. By default, this part is visible.
exThumbPart	256	The thumb part or the scroll box region. By default, the thumb is visible.
exUpperBackPart	128	The area between the thumb and the right/bottom button. By default, this part is visible.
exBackgroundPart	640	The union between the exLowerBackPart and the exUpperBackPart parts. By default, this part is visible.
exRightBPart	64	(>) The right or down button. By default, this button is visible.
exRightB1Part	32	(R1) The first additional button in the right or down side. By default, this button is hidden.

exRightB2Part	16	(R2) The second additional button in the right or down side. By default, this button is hidden.
exRightB3Part	8	(R3) The third additional button in the right or down side. By default, this button is hidden.
exRightB4Part	4	(R4) The forth additional button in the right or down side. By default, this button is hidden
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 ZoomModeEnum

Specifies the way how the chart is zoomed. Use the ZoomWidthMode and ZoomHeightMode properties to zoom the chart.

Name	Value	Description
exDefaultSize	0	The chart is not zoomed.
exCustomSize	1	The user specifies the factor zoom. Use the ZoomWidth and ZoomHeight property specify the factor zoom on x or y axis.
exControlSize	2	The control zooms the chart to fit the control's area.

Appearance object

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

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

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

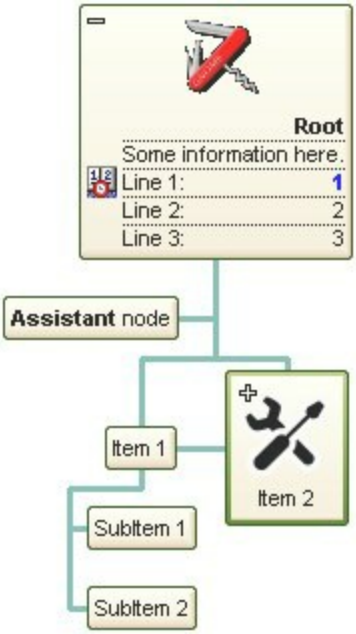
Adds or replaces a skin object to the control.

Type	Description
ID as Long	A Long expression that indicates the index of the skin being added or replaced. The value must be between 1 and 126, so Appearance collection should holds no more than 126 elements.
Skin as Variant	A string expression that indicates one of the following: • 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 "CP:n l t r b", 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. For instance, the "**CP:1 4 0 -4 0**", indicates that the skin is displayed on a smaller rectangle like follows. • 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

A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. You can use this option when using the EBN file directly in the resources of the project. For instance, the VB6 provides the LoadResData to get the safe array o bytes for specified resource, while in VB/NET or C# the internal class Resources provides definitions for all files being inserted. (ResourceManager.GetObject("ebn", resourceCulture)).

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

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



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

- selected node, SelColor property
- borders of the nodes, Background property
- the background for all nodes, BackColorNode property
- node's background, BackColor property
- control's border, using the Appearance property.

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

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

```
With ChartView1
   .VisualAppearance.Add 1, "D:\Temp\ExOrgChart.Help\select.ebn"
   .SelColor = &H1000000
End With
```

The following C++ sample changes the visual appearance for the selected node:

```
#include "Appearance.h"
m_chartview.GetVisualAppearance().Add( 1,
COleVariant("D:\\Temp\\ExOrgChart.Help\\select.ebn") );
m_chartview.SetSelColor( 0x1000000 );
```

The following VB.NET sample changes the visual appearance for the selected node:

```
With AxChartView1
   .VisualAppearance.Add(1, "D:\Temp\ExOrgChart.Help\select.ebn")
   .Template = "SelColor = 16777216"
End With
```

The following C# sample changes the visual appearance for the selected node:

```
axChartView1.VisualAppearance.Add(1, "D:\\Temp\\ExOrgChart.Help\\select.ebn");
```

```
axChartView1.Template = "SelColor = 16777216";
```

The following VFP sample changes the visual appearance for the selected node:

```
With thisform.ChartView1
   .VisualAppearance.Add(1, "D:\Temp\ExOrgChart.Help\select.ebn")
   .SelColor = 16777216
EndWith
```

The screen shot was generated using the following template:

```
BeginUpdate
VisualAppearance
{
   Add( 1,
"gBFLBCJwBAEHhEJAEGg4BU4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK
 )
   Add( 2,
"gBFLBCJwBAEHhEJAEGg4BSoGg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK
 )
}
Background(0) = 16777216
SelColor = 33554432
HideSelection = False

HasButtons = 3
ButtonsAlign = 0
PenWidthLink = 3
Root
{
   Caption = "Root Some information here.
Line 1:1
Line 2:2
Line 3:3"
   Image = 1
   Picture =
"gBHJJGHA5MIwAEIe4AAAFhwFBwOCERDYXC4bEAgEopFIwiwwjgwGQyHcRHcZHcjHcrHZE
```

```
   AddAssistant("Assistant node")
}
Nodes
{
   Add("Item 1",,"Key1")
   {
      HasButton = False
      LinkTo = "Key2"
   }
   Add("SubItem 1","Key1")
   Add("SubItem 2","Key1")
   Add("Item 2",,"Key2")
   {
      Expanded = False
      ArrangeSiblingNodesAs = 1
      Picture =
"gBHJJGHA5MIwAEIe4AAAFhwbiAliQwig7ixFjBQjRbjhljxwkB7kSFkiQkybICllSwli7lzFmDQmT

   }
   Add("SubItem 1","Key2")
   Add("SubItem 2","Key2")
}
EndUpdate
```

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:

- selected node, SelColor property
- borders of the nodes, Background property
- the background for all nodes, BackColorNode property
- node's background, BackColor property

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

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

- selected node, SelColor property
- borders of the nodes, Background property
- the background for all nodes, BackColorNode property
- node's background, BackColor property

property Appearance.RenderType as Long

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

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

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

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

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

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

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

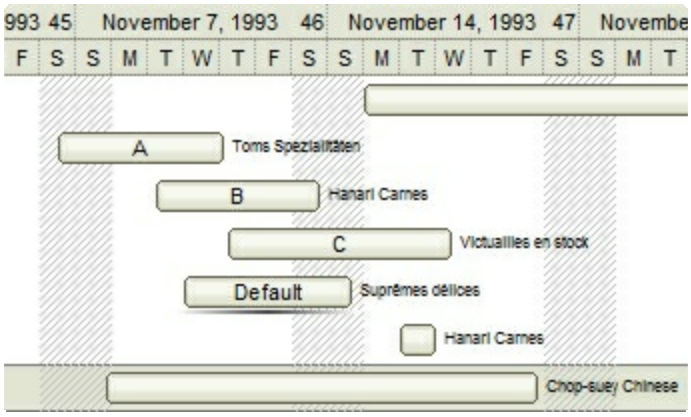
In the following screen shot the following objects displays the current EBN with a different color:

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

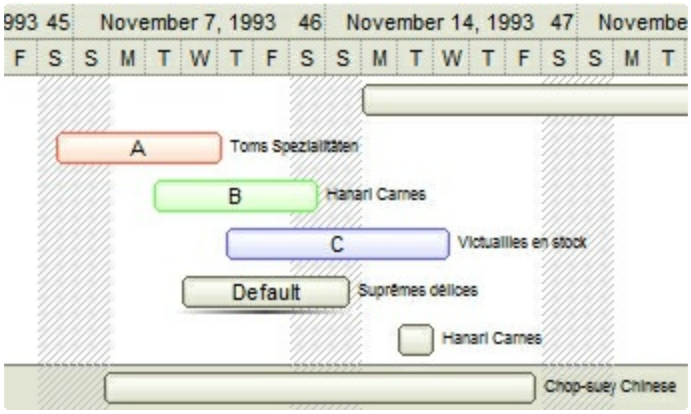
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

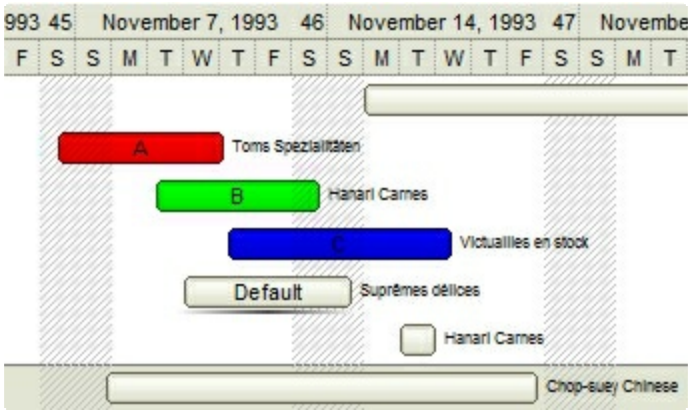
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



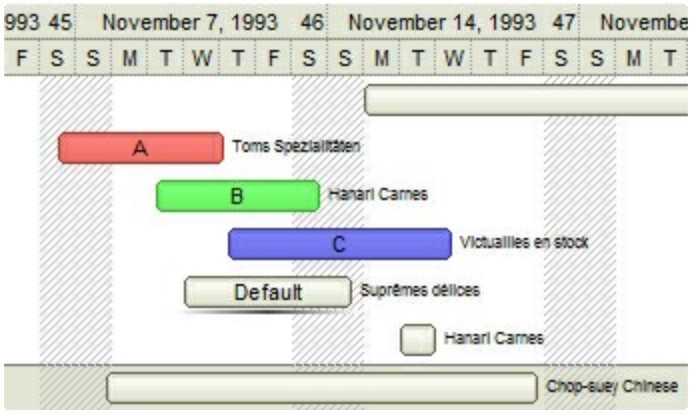
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the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



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

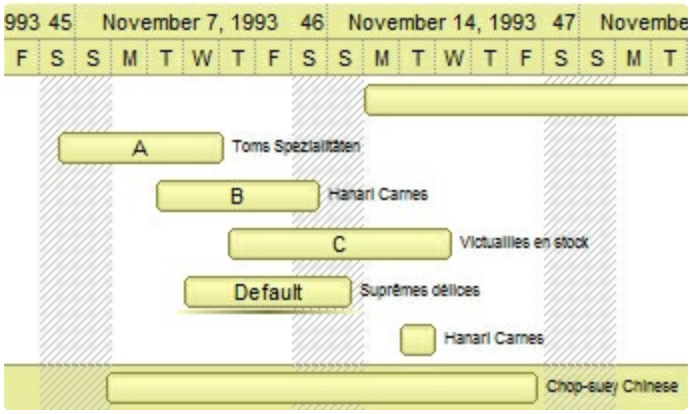


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

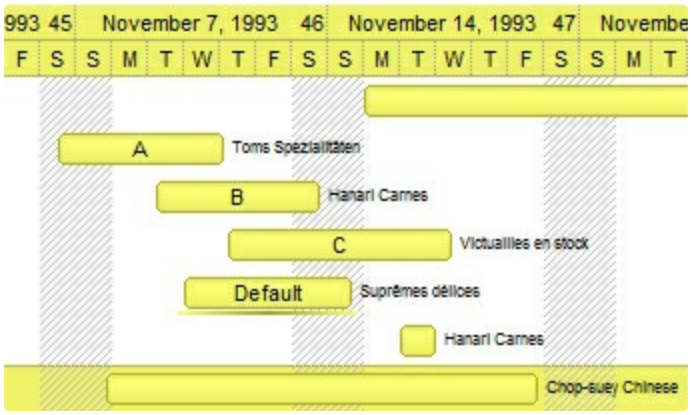


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

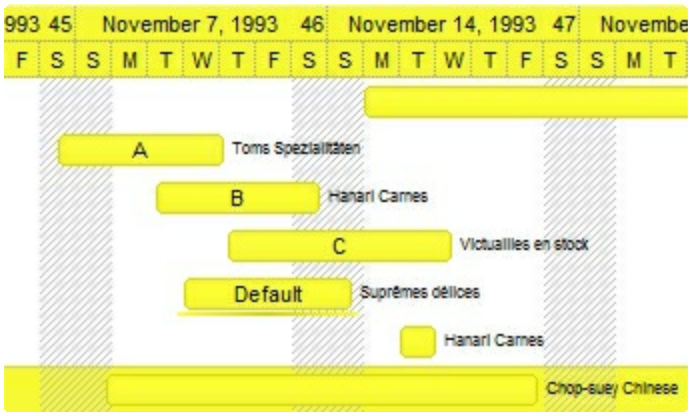
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



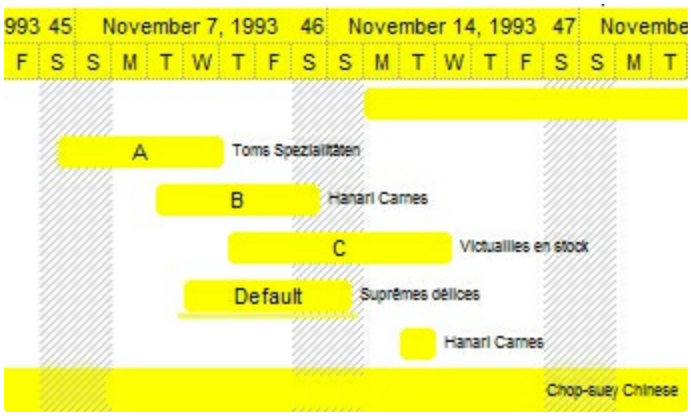
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



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



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


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

ChartView 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: {F4DFE455-01FE-420E-A088-64346DCC3791}. The object's program identifier is: "Exontrol.ChartView". The /COM object module is: "ExOrgChart.dll"

The ExOrgChart component handles and displays organigrams. Use the Nodes property to access the control's Nodes collection. 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. Use the VisualAppearance property to add new skins to the control. The control supports the following properties and methods:

Name	Description
AcceptFiles	Specifies whether the control accepts drag-and-drop files.
AllowCopyTemplate	Specifies whether the Shift + Ctrl + Alt + Insert sequence copies the control's content to the clipboard, in template form.
AllowEdit	Specifies the combination of keys that allows the user to edit a node.
AllowMoveChart	Indicates the combination of keys so the user moves or scrolls the chart at runtime.
AllowMoveNode	Specifies the combination of keys the user can move a node at runtime.
AllowResizeChart	Specifies the keys combination so the user can magnify or shrink the chart at runtime (zooming).
AllowSelectNothing	Specifies whether the user can select nothing if he clicks out of any node.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
AntiAliasing	Specifies whether smoothing (antialiasing) is applied to lines or curves of the objects in the control.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
BackColor	Specifies the control's background color.
BackColorNode	Specifies the default node's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
	Maintains performance when items are added to the

BeginUpdate	control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BorderHeight	Sets or retrieves a value that indicates the border height of the control.
BorderWidth	Sets or retrieves a value that indicates the border width of the control.
ButtonsAlign	Specifies the alignment of the +/- buttons.
ChartHeight	Retrieves the height in pixels to display the entire chart.
ChartWidth	Retrieves the width in pixels to display the entire chart.
Copy	Copies the control's content to the clipboard in EMF format.
CopyTo	Exports the control's view to an EMF file.
Cursor	Gets or sets the cursor that is displayed when the mouse pointer hovers the control.
DefaultNodePadding	Returns or sets a value that indicates the padding of the nodes in the control.
DragOutsideDef	Indicates the options to scroll the control's content like speed, step, and so on while user moves a node by drag and drop outside of the control's content.
DrawRoundNode	Specifies a value that indicates whether the node has borders with round corners.
EditNode	Edits the specified node.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EnsureVisibleNode	Ensures the given node is in the visible client area.
EnsureVisibleOnSelect	Retrieves or sets a value that indicates whether the control ensures the selected node is visible.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
ExpandOnDblClk	Expands or collapses a node when the user dbl clicks the node.
ExploreFromNode	Explores the organigram from the node.
	Retrieves or sets a value that indicates whether the height

FixedHeightNode	of the node's caption is fixed.
FixedWidthNode	Retrieves or sets a value that indicates whether the width of the node's caption is fixed.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
ForeColorNode	Specifies the default node's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
FrameFromPoint	Gets the frame from point.
Frames	Gets the control's collection of frames.
HasButtons	Specifies whether a parent node displays +/- buttons if it contains child nodes.
HasButtonsCustom	Specifies the index of icons for +/- signs when the HasButtons property is exCustom.
hEBNList	Retrieves the handle of the skins list.
hIconList	Retrieves the handle of the icons list.
HideSelection	Specifies whether the selection in the control is hidden when the control loses the focus.
hPictureList	Retrieves the handle of the pictures list.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Image List Control.
ImageSize	Retrieves or sets the size of icons the control displays..
IndentChild	Retrieves or sets the amount, in pixels, that child nodes are indented relative to their parent nodes.
IndentSiblingX	Specifies the horizontal distance, in pixels between two siblings node.
IndentSiblingY	Specifies the vertical distance, in pixels between two siblings node.
Layout	Specifies the way the chart displays the diagram.
LinkAssistantColor	Specifies the color for assistant links.

LinkCaptionFromPoint	Gets the node whose caption on the link hovers the specified point.
LinkColor	Specifies the color for links.
LinkToColor	Specifies the color for 'LinkTo' links.
LoadXML	Loads an XML document from the specified location, using MSXML parser.
MaxZoomHeight	Gets or sets a value indicating how large the chart will appear on vertical axis (max value).
MaxZoomWidth	Gets or sets a value indicating how large the chart will appear on horizontal axis (max value).
MinZoomHeight	Gets or sets a value indicating how large the chart will appear on vertical axis (min value).
MinZoomWidth	Gets or sets a value indicating how large the chart will appear on horizontal axis (min value).
NodeFromPoint	Gets the node from point.
Nodes	Gets the control's collection of nodes.
PenBorderNode	Specifies the type of pen used to draw the node's borders.
PenLink	Specifies the type of the pen used to paint the links between nodes.
PenLinkAssistant	Specifies the type of the pen used to paint the links between assistant nodes.
PenLinkTo	Specifies the type of the pen used to show the 'LinkTo' links.
PenWidthLink	Specifies the width of the links between nodes.
PenWidthLinkAssistant	Specifies the width of the links between assistant nodes.
PenWidthLinkTo	Specifies the width of the 'LinkTo' links.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureAspectRatioNode	Specifies the default aspect ratio for the node's picture.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
PictureHeightNode	Specifies the height of the node's picture.
PictureWidthNode	Specifies the width of the node's picture.
Refresh	Refreses the control.

ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
Root	Gets the root node.
SaveXML	Saves the control's content as XML document to the specified location, using the MSXML parser.
ScrollButtonHeight	Specifies the height of the button in the vertical scrollbar.
ScrollButtonWidth	Specifies the width of the button in the horizontal scrollbar.
ScrollByClick	Specifies a value that indicates whether the user scrolls the control's content by clicking the client area.
ScrollFont	Retrieves or sets the scrollbar's font.
ScrollHeight	Specifies the height of the horizontal scrollbar.
ScrollOnCursor	Scrolls the chart as the cursor indicates.
ScrollOnEnsure	Specifies a value that indicates whether the control scrolls the control's content when ensuring that a node is visible.
ScrollOrderParts	Specifies the order of the buttons in the scroll bar.
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.
ScrollPos	Specifies the vertical/horizontal scroll position.
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.
SelColor	Retrieves or sets a value that indicates the color used to mark the selected node.
SelectNode	Specifies the selected node.
ShadowNode	Specifies whether the node has shadow.
ShowAddNew	Specifies whether the selected node shows or hides add new buttons.

ShowAssistants	Retrieves or sets a value that indicates whether the assistant nodes are shown.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowLinksDir	Specifies whether links show the direction.
ShowRoundLink	Specifies whether the round links are shown between parent and child nodes.
ShowToolTip	Shows the specified 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.
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.
ToTemplate	Generates the control's template.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.
WidthNode	Specifies the maximum width of the nodes captions.
ZoomHeight	Gets or sets a value indicating how large the chart will appear on vertical axis.
ZoomHeightMode	Specifies a value that indicates whether the ZoomHeight property is updated when the control is resized.
ZoomWidth	Gets or sets a value indicating how large the chart will appear on horizontal axis.
ZoomWidthMode	Specifies a value that indicates whether the ZoomWidth property is updated when the control is resized.

property ChartView.AcceptFiles as Boolean

Specifies whether the control accepts drag-and-drop files.

Type	Description
Boolean	A Boolean expression that specifies whether the control accepts files by drag and drop.

By default, the AcceptFiles property is False. If the AcceptFiles property is True, the control fires the DropFile event once the user drags a file over a node.

property ChartView.AllowCopyTemplate as Boolean

Specifies whether the Shift + Ctrl + Alt + Insert sequence copies the control's content to the clipboard, in template form.

Type	Description
Boolean	A Boolean expression that indicates whether the Shift + Ctrl + Alt + Insert sequence copies the control's content to the clipboard, in template form.

By default, the AllowCopyTemplate property is True, only for trial-demo version, and False, for the registered version. So, by default, the Shift + Ctrl + Alt + Insert sequence is working in the trial version, and it doesn't work on the registered version. Use the Version property to find out what version of the control you are running. Use the AllowCopyTemplate property for debugging purpose. Use the AllowCopyTemplate property to easily copy the control's content to the clipboard, as template form, and so you can send us a sample without being necessary to send the entire sample to us. The AllowCopyTemplate property is not serialized in the form's persistence, so you need to set it in the code for a particular value. If the AllowCopyTemplate property is True, the user may use the Shift + Ctrl + Alt + Insert sequence to copy the control's content to the clipboard, in template form. If the control manages to copy the control's content to the clipboard, you should hear a beep. The property uses the ToTemplate property to generate the control's template, at runtime. The format of the clipboard being copied is plain text. Use the Template property to apply the generated template to an empty control.

property ChartView.AllowEdit as AllowKeysEnum

Specifies the combination of keys that allows the user to edit a node.

Type	Description
AllowKeysEnum	An AllowKeyEnum expression that specifies the key combination so the user can edit the node's caption.

By default, the AllowEdit is set on exLeftClick + exDblLick, which means the user is able to edit the node's caption by double clicking the left mouse button. Use the AllowEdit property to prevent editing the node's caption at runtime, or to specify a different keys combination so the user can edit the node's caption. The LayoutStartChanging(exEditNode) event notifies your application once the user starts editing the node's caption. Use the NodeFromPoint(-1,-1) property to get the node from the current position. The LayoutEndChanging(exEditNode) event notifies your application once of the edit operation ends. The Caption property indicates the caption of the node being edited. Use the Editable property to specify whether a specified node can be edited at runtime. The Background(exEditNodeBackColor)/Background(exEditNodeForeColor) property specifies the background/foreground color of the edit field being displayed on the node while editing. Use the EditNode to programmatically edit the giving node.

property ChartView.AllowMoveChart as AllowKeysEnum

Indicates the combination of keys so the user moves or scrolls the chart at runtime.

Type	Description
AllowKeysEnum	An AllowKeysEnum expression that specifies the keys combination so the user can scroll the control's content without using the control's scroll bars.

By default, the AllowMoveChart property is exLeftClick. This means that if you single click the left mouse button and start dragging the control's context is scrolled to the dragging direction. This property has effect only, if the control's scroll bars are visible, in other words, there are nodes that are not visible in the control's client area. The AllowMoveNode property specifies the combination of keys so the user can move a node from one parent to another. The AllowResizeNode property specifies the combination of keys so the user can resize the chart at runtime. The ScrollPos property specifies the control's scroll position.

property ChartView.AllowMoveNode as AllowKeysEnum

Specifies the combination of keys the user can move a node at runtime.

Type	Description
AllowKeysEnum	An AllowKeysEnum expression that specifies the keys combination so the user can move a node from one parent to another.

By default, the AllowMoveNode property is exLeftClick + exSHIFTKey. This means that if you single click the left mouse button on the node while the SHIFT key is pressed, you can drag the node to a new parent. You can use the NodeFromPoint property to get the node from the current position. The Parent property specifies the parent node. The LayoutStartChanging(exMoveNode) event notifies your application once the user starts dragging a node. The LayoutEndChanging(exMoveNode) event notifies your application once user drops the node to a new position. The DragOutsideDef property customizes the speed to scroll the control's content while user moves nodes by drag and drop then outside of the control's client area.

property ChartView.AllowResizeChart as AllowKeysEnum

Specifies the keys combination so the user can magnify or shrink the chart at runtime (zooming).

Type	Description
AllowKeysEnum	An AllowKeysEnum expression that specifies the keys combination so the user can resize the control's content at runtime.

By default, the AllowResizeChart property is exMiddleClick, which means that once the user clicks the middle mouse button, the user can resize the chart by dragging. The ZoomWidth property specifies a value that indicates how large the chart will appear on horizontal axis. The LayoutStartChanging(exResizeChart) event notifies your application once the user starts resizing the chart. The LayoutEndChanging(exResizeChart) event notifies your application once the chart is resized. The ZoomHeight property specifies a value that indicates how large the chart will appear on vertical axis. Use the MinZoomWidth/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. Use the MinZoomHeight/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. The AllowMoveNode property specifies the combination of keys so the user can move a node from one parent to another.

property ChartView.AllowSelectNothing as Boolean

Specifies whether the user can select nothing if he clicks out of any node.

Type	Description
Boolean	A Boolean expression that specifies whether the user can select nothing if the user clicks outside of the node.

By default, the AllowSelectNothing property is False. The AllowSelectNothing property allows selecting nothing if the user clicks outside the node. Use the SelectNode property to specify the selected node.

property ChartView.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 ChartView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   With ChartView1
      .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 AxChartView1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXCHARTVIEWLib._IChartViewEvents_MouseMoveEvent) Handles AxChartView1.MouseMoveEvent
   With AxChartView1
      .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 axChartView1_MouseMoveEvent(object sender,
AxEXCHARTVIEWLib._IChartViewEvents_MouseMoveEvent e)
{
   axChartView1.ShowToolTip(axChartView1.get_AnchorFromPoint(-1, -1));
}
```

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

```
void OnMouseMoveChartView1(short Button, short Shift, long X, long Y)
{
   COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
   m_chartView.ShowToolTip( m_chartView.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 .ChartView1
      .ShowToolTip(.AnchorFromPoint(-1, -1))
   EndWith
endwith
```

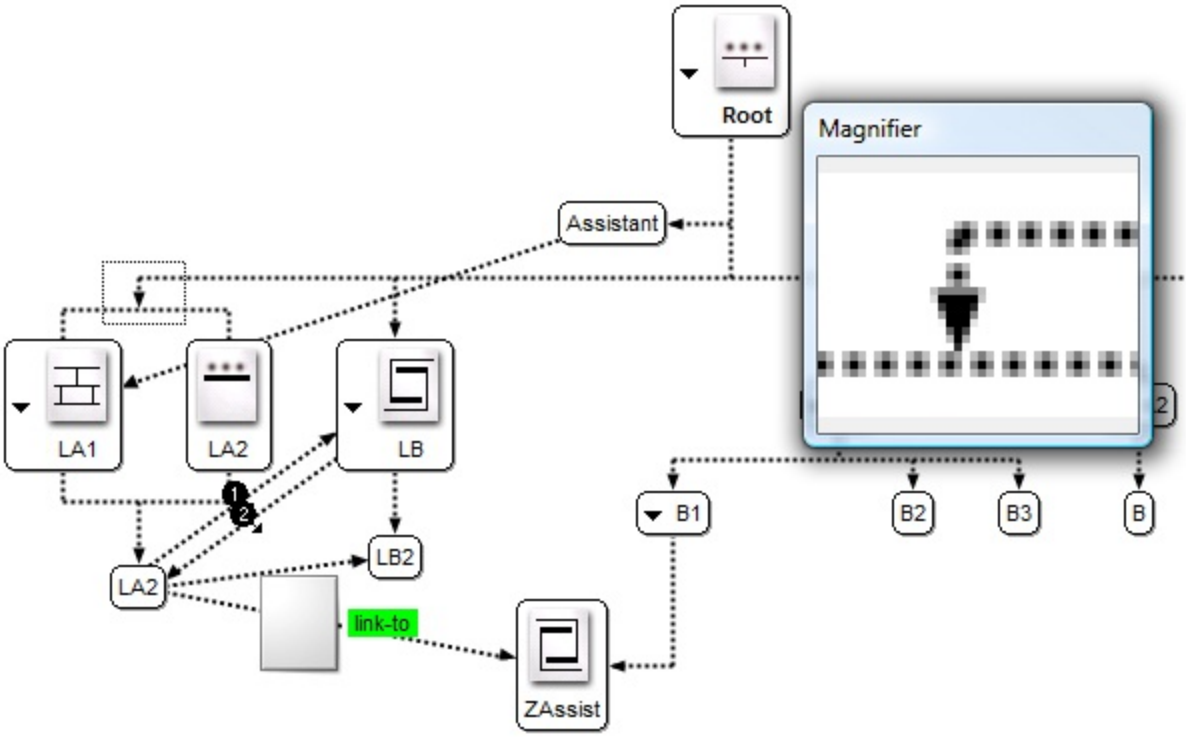
property ChartView.AntiAliasing as Boolean

Specifies whether smoothing (antialiasing) is applied to lines or curves in the control.

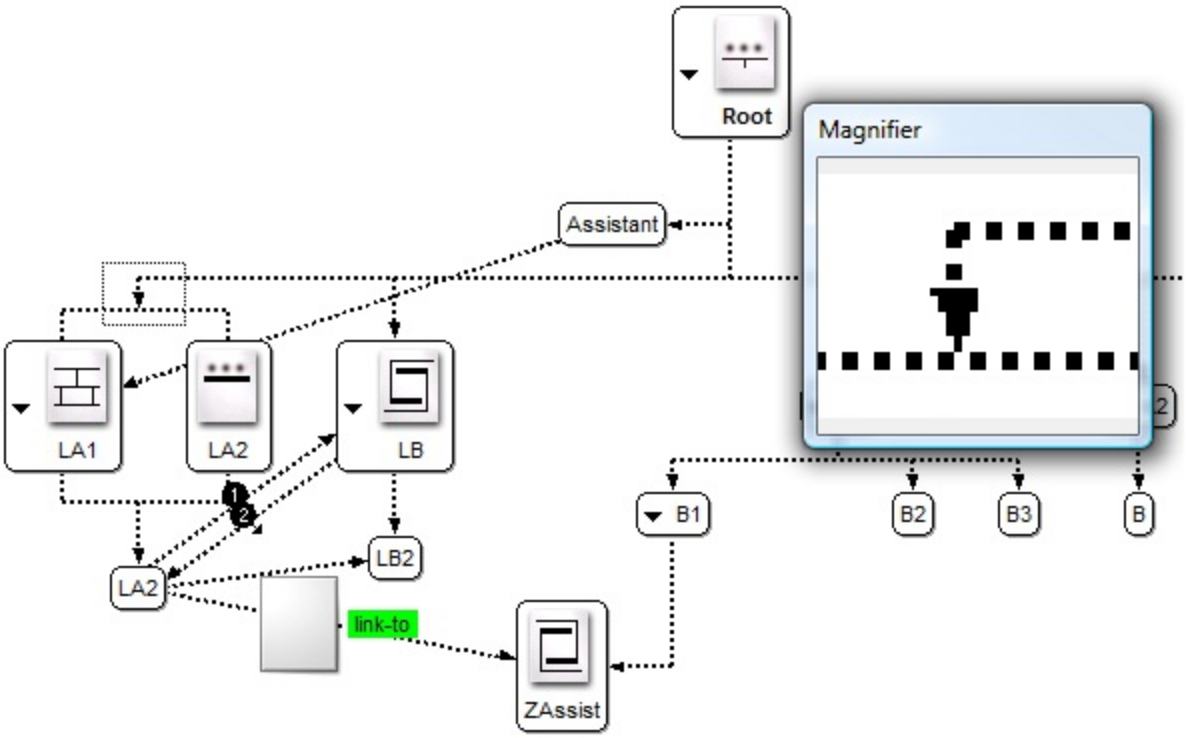
Type	Description
Boolean	A Boolean expression that specifies whether the control uses the antialiasing rendering to show the lines or curves in the control.

By default, the AntiAliasing property is False. In other words, the AntiAliasing property determines the rendering quality for different objects in the control. For instance, you can use the antialiasing feature to show the arrows or links between nodes more smoothly. The ShowLinksDir property specifies whether the links show the directions. Use the LinkTo property to link arbitrary a node with another. The LinkToCaption property specifies the HTML caption being shown on a link.

The following figure illustrates the visual distortion that occurs when anti-aliasing is used.



The following figure illustrates the visual distortion that occurs when anti-aliasing is not used.

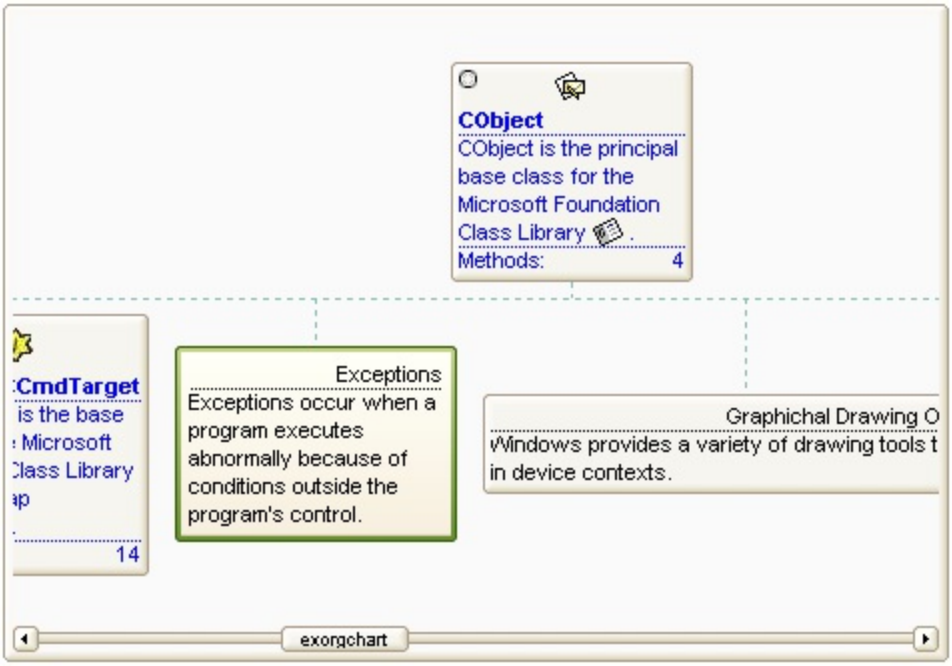

Root
Magnifier
Assistant
LA1
LA2
LB
B1
B2
B3
B
LA2
LB2
link-to
ZAssist

property ChartView.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 chart, scrollbars are always shown in the control's client area. The skin may contain transparent objects, and so you can define round corners. The normal.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file***

Use the Appearance property to remove the control's borders, or to define new type of borders. By default, the Appearance property is Sunken. Use the BackColor property to specify the control's background color. Use the Picture property to display a picture on the control's background. Use the PenLink property to draw links between parent and child nodes. Use the PenWidthLink property to specify the width of the link between nodes. 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 ChartView1
   .BeginUpdate
      .VisualAppearance.Add &H16, "c:\temp\normal.ebn"
      .Appearance = &H16000000
      .BackColor = RGB(250, 250, 250)
   .EndUpdate
End With
```

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

```
With AxChartView1
   .BeginUpdate()
   .VisualAppearance.Add(&H16, "c:\temp\normal.ebn")
   .Appearance = &H16000000
   .BackColor = Color.FromArgb(250, 250, 250)
   .EndUpdate()
End With
```

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

```
axChartView1.BeginUpdate();
axChartView1.VisualAppearance.Add(0x16, "c:\\temp\\normal.ebn");
axChartView1.Appearance = (EXCHARTVIEWLib.AppearanceEnum)0x16000000;
axChartView1.BackColor = Color.FromArgb(250, 250, 250);
axChartView1.EndUpdate();
```

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

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

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

```
with thisform.ChartView1
```

```
  .BeginUpdate
    .VisualAppearance.Add(0x16, "c:\temp\normal.ebn")
    .Appearance = 0x16000000
    .BackColor = RGB(250, 250, 250)
  .EndUpdate
endwith
```

method ChartView.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 ChartView1_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 ChartView.BackColor as Color

Specifies the control's background color.

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

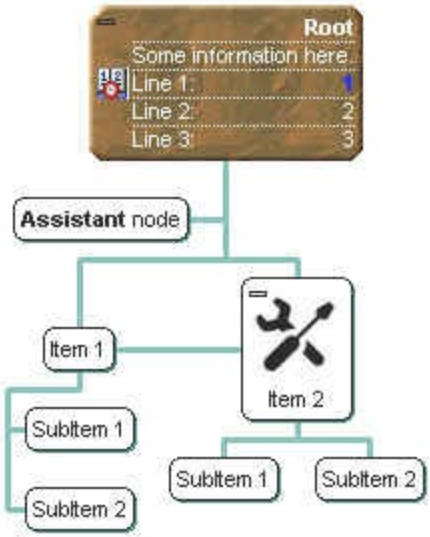
Use the BackColor property to define the control's background color. Use the BackColorNode property to define the default background color for nodes. Use the BackColor property to define the background color for a specified node. Use the Picture property to display a picture on the control's background. Use the SelColor property to specify the color to mark the selected node. Use the SelectNode property to specify the selected node. Use the BackColor property to specify the control's background color. Use the LinkColor property to specify the color for the links between nodes. For instance, the SelColor property has the same value as BackColor property, the control doesn't paint the mark around the selected node.

property ChartView.BackColorNode as Color

Specifies the default node's background color.

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

Use the BackColorNode property to define the default background color for nodes. Use the BackColor property to define the background color for a specific node. Use the ClearBackColor method to clear the node's background color. Use the BackColor property to specify the control's background color. Use the Picture property to display a picture on the control's background. Use the SelColor property to specify the color to mark the selected node. Use the ForeColorNode property to specify the foreground color for all nodes. The DefaultNodePadding property defines the padding for all nodes.

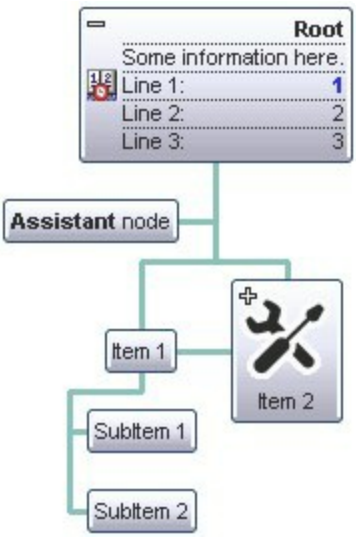


property ChartView.Background(Part as BackgroundPartEnum) as Color

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

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

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



The following VB sample changes the frame for all nodes. The sample applies the "" frame to all nodes.

```
With ChartView1
  .VisualAppearance.Add 2, "D:\Temp\ExOrgChart.Help\node.ebn"
  .Background(exNodeFrame) = &H2000000
```

```
End Wit
```

The following C++ sample changes the frame for all nodes:

```
#include "Appearance.h"
m_chartview.GetVisualAppearance().Add( 2,
COleVariant("D:\\Temp\\ExOrgChart.Help\\node.ebn") );
m_chartview.SetBackground( 0, 0x2000000 );
```

The following VB.NET sample changes the frame for all nodes:

```
With AxChartView1
   .VisualAppearance.Add(2, "D:\Temp\ExOrgChart.Help\node.ebn")
   .set_Background(EXORGCHARTLib.BackgroundPartEnum.exNodeFrame, &H2000000)
End With
```

The following C# sample changes the frame for all nodes:

```
axChartView1.VisualAppearance.Add(2, "D:\\Temp\\ExOrgChart.Help\\node.ebn");
axChartView1.set_Background(EXORGCHARTLib.BackgroundPartEnum.exNodeFrame,
0x2000000);
```

The following VFP sample changes the frame for all nodes:

```
With thisform.ChartView1
   .VisualAppearance.Add(2, "D:\Temp\ExOrgChart.Help\node.ebn")
   .Background(0) = 33554432
EndWith
```

where the 33554432 in hexa is 0x2000000

method ChartView.BeginUpdate ()

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

Type	Description

The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Use the Add method to add new child nodes. Use the Remove method to remove a node from the control. Use the Root property to get the root node of the control.

The following VB sample adds four nodes to the control :

```
With ChartView1
   .BeginUpdate
   With .Nodes
      .Add "Item 1", "root", "Key1"
      .Add "Item 2", "root"
      .Add "Sub Item 1", "Key1"
      .Add "Sub Item 2", "Key1"
   End With
   .EndUpdate
End With
```

The following C++ sample adds four nodes to the control :

```
#include "nodes.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
m_chartview.BeginUpdate();
CNodes nodes = m_chartview.GetNodes();
nodes.Add( "Item 1", COleVariant( "root" ), COleVariant("Key1"), vtMissing, vtMissing );
nodes.Add( "Item 2", COleVariant( "root" ), vtMissing, vtMissing, vtMissing );
nodes.Add( "Sub Item 1", COleVariant( "Key1" ), vtMissing, vtMissing, vtMissing );
nodes.Add( "Sub Item 2", COleVariant( "Key1" ), vtMissing, vtMissing, vtMissing );
m_chartview.EndUpdate();
```

The following VB.NET sample adds four nodes to the control :

```
With AxChartView1
   .BeginUpdate()
   With .Nodes
```

```
        .Add("Item 1", "root", "Key1")
        .Add("Item 2", "root")
        .Add("Sub Item 1", "Key1")
        .Add("Sub Item 2", "Key1")
    End With
    .EndUpdate()
End With
```

The following C# sample adds four nodes to the control :

```
axChartView1.BeginUpdate();
EXORGCHARTLib.Nodes nodes = axChartView1.Nodes;
nodes.Add("Item 1", "root", "Key1", null, null);
nodes.Add("Item 2", "root", null, null, null);
nodes.Add("Sub Item 1", "Key1", null, null, null);
nodes.Add("Sub Item 2", "Key1", null, null, null);
axChartView1.EndUpdate();
```

The following VFP sample adds four nodes to the control :

```
With thisform.ChartView1
    .BeginUpdate
    With .Nodes
        .Add("Item 1", "root", "Key1")
        .Add("Item 2", "root")
        .Add("Sub Item 1", "Key1")
        .Add("Sub Item 2", "Key1")
    EndWith
    .EndUpdate
EndWith
```

property ChartView.BorderHeight as Long

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

Type	Description
Long	A long expression that defines the height of the top and bottom control's border. in pixels.

By default, the BorderHeight property is 4 pixels. The control's client area excludes the size of the borders. Use the BorderWidth and BorderHeight properties to define the size of control's borders. The BorderHeight property is specified in pixels. Use the FixedHeightNode property to specify the size of nodes. Use the Font property to specify the control's font.

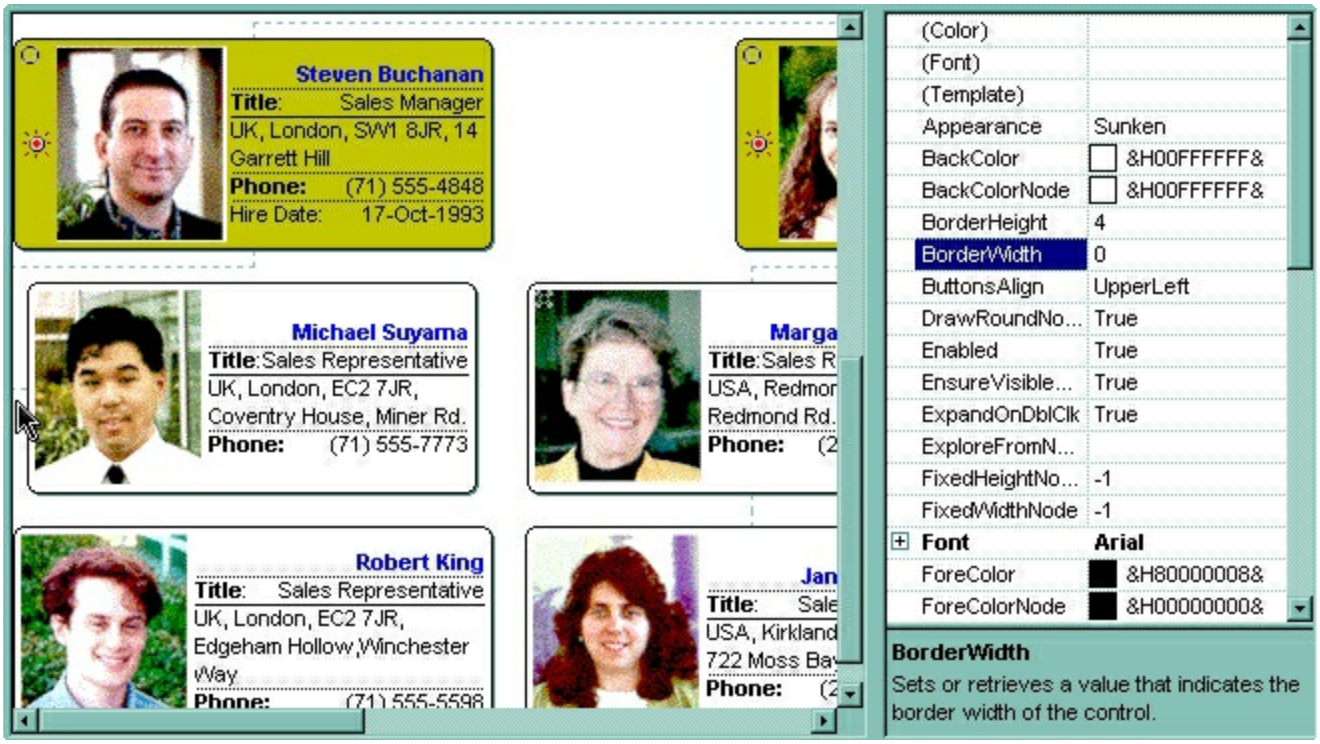
property ChartView.BorderWidth as Long

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

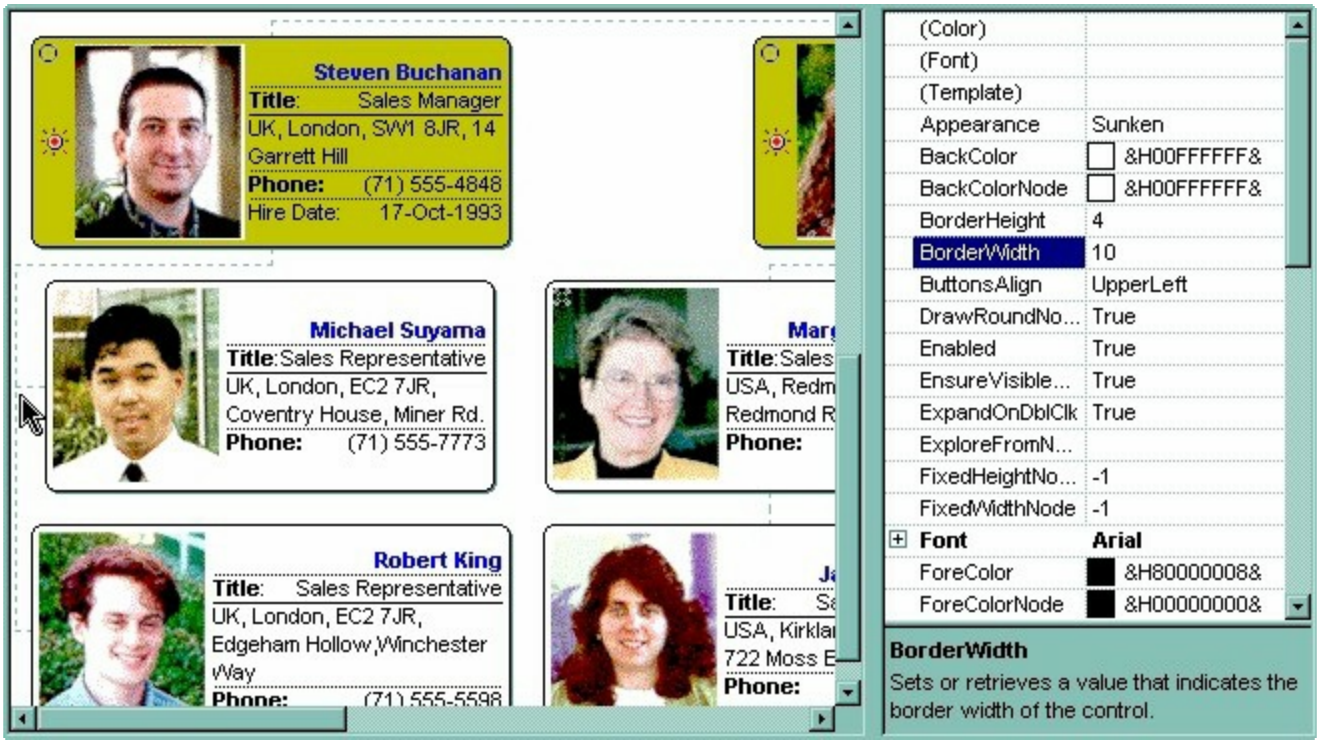
Type	Description
Long	A long expression that specifies the width of the left and right control's borders, in pixels.

The control's client area excludes the size of the borders. Use the BorderWidth and BorderHeight properties to define the size of control's borders. The BorderWidth property is specified in pixels. By default, the BorderWidth property is 10 pixels. Use the FixedWidthNode property to specify the size of nodes. Use the Font property to specify the control's font. The ChartHeight property gets the height in pixels required to display the entire chart. The ChartWidth property gets the width in pixels required to display the entire chart.

The following screen shot shows the chart when the BorderWidth property is 0. (Look on the left side of the chart).



The following screen shot shows the chart when the BorderWidth property is 10. (Look on the left side of the chart).


Steven Buchanan
Title: Sales Manager
UK, London, SW1 8JR, 14 Garrett Hill
Phone: (71) 555-4848
Hire Date: 17-Oct-1993
Michael Suyama
Title: Sales Representative
UK, London, EC2 7JR, Coventry House, Miner Rd.
Phone: (71) 555-7773
Robert King
Title: Sales Representative
UK, London, EC2 7JR, Edgeham Hollow, Winchester Way
(Color)
(Font)
(Template)
Appearance Sunken
BackColor &H00FFFFFF&
BackColorNode &H00FFFFFF&
BorderHeight 4
BorderWidth 10
ButtonsAlign UpperLeft
DrawRoundNo... True
Enabled True
EnsureVisible... True
ExpandOnDblClk True
ExploreFromN...
FixedHeightNo... -1
FixedWidthNode -1
Font Arial
ForeColor &H80000008&
ForeColorNode &H00000000&
BorderWidth
Sets or retrieves a value that indicates the border width of the control.

property ChartView.ButtonsAlign as PictureDisplayEnum

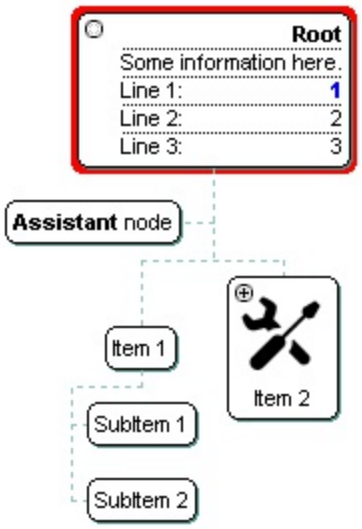
Specifies the alignment of the +/- buttons.

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the position of +/- buttons inside nodes. The Tile, Stretch values have no effect.

By default, the ButtonsAlign property is UpperLeft. Use the ButtonsAlign property to specify the position where the +/- buttons are displayed on nodes. Use the HasButtons property to display the +/- buttons for nodes that contain child nodes. Use the HasButton property to hide the +/- button inside a particular node.

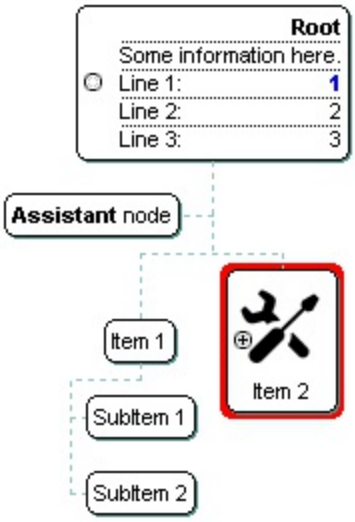
The following VB sample displays the +/- signs on the upper left corner:

```
With ChartView1
   .HasButtons = exCircle
   .ButtonsAlign = UpperLeft
End With
```



The following VB sample displays the +/- signs on the middle left border:

```
With ChartView1
   .HasButtons = exCircle
   .ButtonsAlign = MiddleLeft
End With
```


Root
Some information here.
Line 1: 1
Line 2: 2
Line 3: 3
Assistant node
Item 1
Item 2
SubItem 1
SubItem 2

property ChartView.ChartHeight as Long

Retrieves the height in pixels to display the entire chart.

Type	Description
Long	A long expression that specifies the height in pixels required to display the entire chart.

The ChartHeight property gets the height in pixels required to display the entire chart. The ChartHeight property does not retrieve the size of the borders. The ChartWidth property specifies the width in pixels required to display the entire chart. Use the Appearance property to remove the control's appearance. Use the BorderWidth property to specify the width in pixels of the control's empty border (in the left or right side). Use the BorderHeight property to specify the height in pixels of the control's empty border (in the top or bottom side). The ChartWidth and ChartHeight properties does NOT include the size of the control's borders (if the Appearance property is not 0). Use the Appearance property on 0, or add 4 pixels, if you are using the Appearance property on not zero.

The following VB sample resizes the control so the entire chart is display (no scroll bars are displayed):

```
Private Sub autoSize()
   With ChartView1
      Dim nMode As ZoomModeEnum
      nMode = .ZoomWidthMode
      .BeginUpdate
         .Width = (2 * .BorderWidth + .ChartWidth) * Screen.TwipsPerPixelX
         .Height = (2 * .BorderHeight + .ChartHeight) * Screen.TwipsPerPixelY
         .ZoomWidthMode = exControlSize
         .ZoomHeightMode = exControlSize
      .EndUpdate
      .ZoomWidthMode = nMode
      .ZoomHeightMode = nMode
   End With
End Sub
```

The following VB.NET sample resizes the control so the entire chart is display (no scroll bars are displayed):

```
Private Sub ASize()
   With Chartview1
```

```
    Dim nMode As exontrol.EXORGCHARTLib.ZoomModeEnum = .ZoomWidthMode
    .BeginUpdate()
    .Width = 2 * Chartview1.BorderWidth + Chartview1.ChartWidth
    .Height = 2 * Chartview1.BorderHeight + Chartview1.ChartHeight
    .ZoomWidthMode = exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize
    .ZoomHeightMode = exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize
    .EndUpdate()
    .ZoomWidthMode = nMode
    .ZoomHeightMode = nMode
  End With
End Sub
```

The following C# sample resizes the control so the entire chart is display (no scroll bars are displayed):

```
private void autoSize()
{
  exontrol.EXORGCHARTLib.ZoomModeEnum nMode = chartview1.ZoomWidthMode;
  chartview1.BeginUpdate();
  chartview1.Width = 2 * chartview1.BorderWidth + chartview1.ChartWidth;
  chartview1.Height = 2 * chartview1.BorderHeight + chartview1.ChartHeight;
  chartview1.ZoomWidthMode =
exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize;
  chartview1.ZoomHeightMode =
exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize;
  chartview1.EndUpdate();
  chartview1.ZoomWidthMode = nMode;
  chartview1.ZoomHeightMode = nMode;
}
```

property ChartView.ChartWidth as Long

Retrieves the width in pixels to display the entire chart.

Type	Description
Long	A long expression that specifies the width in pixels required to display the entire chart.

The ChartWidth property gets the width in pixels required to display the entire chart. The ChartWidth property does not retrieve the size of the borders. The ChartHeight property retrieves the height in pixels to display the entire chart. Use the Appearance property to remove the control's appearance. Use the BorderWidth property to specify the width in pixels of the control's empty border (in the left or right side). Use the BorderHeight property to specify the height in pixels of the control's empty border (in the top or bottom side). The ChartWidth and ChartHeight properties does NOT include the size of the control's borders (if the Appearance property is not 0). Use the Appearance property on 0, or add 4 pixels, if you are using the Appearance property on not zero.

The following VB sample resizes the control so the entire chart is display (no scroll bars are displayed):

```
Private Sub autoSize()
   With ChartView1
      Dim nMode As ZoomModeEnum
      nMode = .ZoomWidthMode
      .BeginUpdate
         .Width = (2 * .BorderWidth + .ChartWidth) * Screen.TwipsPerPixelX
         .Height = (2 * .BorderHeight + .ChartHeight) * Screen.TwipsPerPixelY
         .ZoomWidthMode = exControlSize
         .ZoomHeightMode = exControlSize
      .EndUpdate
      .ZoomWidthMode = nMode
      .ZoomHeightMode = nMode
   End With
End Sub
```

The following VB.NET sample resizes the control so the entire chart is display (no scroll bars are displayed):

```
Private Sub ASize()
   With Chartview1
```

```
    Dim nMode As exontrol.EXORGCHARTLib.ZoomModeEnum = .ZoomWidthMode
    .BeginUpdate()
    .Width = 2 * Chartview1.BorderWidth + Chartview1.ChartWidth
    .Height = 2 * Chartview1.BorderHeight + Chartview1.ChartHeight
    .ZoomWidthMode = exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize
    .ZoomHeightMode = exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize
    .EndUpdate()
    .ZoomWidthMode = nMode
    .ZoomHeightMode = nMode
  End With
End Sub
```

The following C# sample resizes the control so the entire chart is display (no scroll bars are displayed):

```
private void autoSize()
{
  exontrol.EXORGCHARTLib.ZoomModeEnum nMode = chartview1.ZoomWidthMode;
  chartview1.BeginUpdate();
  chartview1.Width = 2 * chartview1.BorderWidth + chartview1.ChartWidth;
  chartview1.Height = 2 * chartview1.BorderHeight + chartview1.ChartHeight;
  chartview1.ZoomWidthMode =
exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize;
  chartview1.ZoomHeightMode =
exontrol.EXORGCHARTLib.ZoomModeEnum.exControlSize;
  chartview1.EndUpdate();
  chartview1.ZoomWidthMode = nMode;
  chartview1.ZoomHeightMode = nMode;
```

method ChartView.Copy ()

Copies the control's content to the clipboard in EMF format.

Type	Description

Use the Copy method to copy the control's content to the clipboard. You can paste this to Microsoft Word, Excel, and so on. By default, the control copies its content to the clipboard when user presses the CTRL + C combination. Use the CopyTo method to export the control's content to an PDF,JPG,PNG,GIF,TIF,BMP,EMF file.

The following VB sample saves the control's content to a file:

```
Clipboard.Clear
ChartView.Copy
SavePicture Clipboard.GetData(), App.Path & "\test.emf"
```

Now, you can open your MS Windows Word application, and you can insert the file using the Insert\Picture\From File menu.

The following C++ function saves the clipboard's data (EMF format) to a picture file:

```
BOOL saveEMFtoFile( LPCTSTR szFileName )
{
   BOOL bResult = FALSE;
   if ( ::OpenClipboard( NULL ) )
   {
      CComPtr spPicture;
      PICTDESC pictDesc = {0};
      pictDesc.cbSizeofstruct = sizeof(pictDesc);
      pictDesc.emf.hemf = (HENHMETAFILE)GetClipboardData( CF_ENHMETAFILE );
      pictDesc.picType = PICTYPE_ENHMETAFILE;
      if ( SUCCEEDED( OleCreatePictureIndirect( &pictDesc;, IID_IPicture, FALSE,
(LPVOID*)&spPicture; ) ) )
      {
         HGLOBAL hGlobal = NULL;
         CComPtr spStream;
         if ( SUCCEEDED( CreateStreamOnHGlobal( hGlobal = GlobalAlloc( GPTR, 0 ), TRUE,
&spStream; ) ) )
         {
```

```
            long dwSize = NULL;
            if ( SUCCEEDED( spPicture->SaveAsFile( spStream, TRUE, &dwSize; ) ) )
            {
                USES_CONVERSION;
                HANDLE hFile = CreateFile( szFileName, GENERIC_WRITE, NULL, NULL,
CREATE_ALWAYS, NULL, NULL );
                if ( hFile != INVALID_HANDLE_VALUE )
                {
                    LARGE_INTEGER l = {NULL};
                    spStream->Seek(l, STREAM_SEEK_SET, NULL);
                    long dwWritten = NULL;
                    while ( dwWritten < dwSize )
                    {
                        unsigned long dwRead = NULL;
                        BYTE b[10240] = {0};
                        spStream->Read( &b;, 10240, &dwRead; );
                        DWORD dwBWritten = NULL;
                        WriteFile( hFile, b, dwRead, &dwBWritten;, NULL );
                        dwWritten += dwBWritten;
                    }
                    CloseHandle( hFile );
                    bResult = TRUE;
                }
            }
        }
    }
    CloseClipboard();
  }
  return bResult;
}
```

The following VB.NET sample copies the control's content to the clipboard (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear()
With AxChartView1
  .Copy()
End With
```

The following C# sample copies the control's content to a file (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear;
axChartView1.Copy();
```

property ChartView.CopyTo (File as String) as Variant

Exports the control's view to an EMF file.

Type	Description
File as String	A String expression that indicates the name of the file to be saved. If present, the CopyTo property retrieves True, if the operation succeeded, else False it is failed. If the File parameter is missing or empty, the CopyTo property retrieves an one dimension safe array of bytes that contains the EMF content. If the File parameter is not empty, the extension (characters after last dot) determines the graphical/ format of the file to be saved as follows: • ***.bmp *.dib *.rle**, saves the control's content in **BMP** format. • ***.jpg *.jpe *.jpeg *.jfif**, saves the control's content in **JPEG** format. • ***.gif**, , saves the control's content in **GIF** format. • ***.tif *.tiff**, saves the control's content in **TIFF** format. • ***.png**, saves the control's content in **PNG** format. • ***.pdf**, saves the control's content to PDF format. The File argument may carry up to 4 parameters separated by the | character in the following order: ***filename.pdf | paper size | margins | options***. In other words, you can specify the file name of the PDF document, the paper size, the margins and options to build the PDF document. By default, the paper size is 210 **mm** × 297 **mm** (A4 format) and the margins are 12.7 **mm** 12.7 **mm** 12.7 **mm** 12.7 **mm**. The units for the paper size and margins can be **pt** for PostScript Points, **mm** for Millimeters, **cm** for Centimeters, **in** for Inches and **px** for pixels. If PostScript Points are used if unit is missing. For instance, 8.27 in x 11.69 in, indicates the size of the paper in inches. Currently, the options can be **single**, which indicates that the control's content is exported to a single PDF page. For instance, the CopyTo("shot.pdf|33.11 in x 46.81 in|0 0 0 0|single") exports the control's content to an A0 single PDF page, with no margins. • ***.emf** or any other extension determines the control to

	save the control's content in **EMF** format. For instance, the CopyTo("c:\temp\snapshot.png") property saves the control's content in PNG format to snapshot.png file.
Variant	A boolean expression that indicates whether the File was successful saved, or a one dimension safe array of bytes, if the File parameter is empty string.

The CopyTo method copies/exports the control's view to BMP, PNG, JPG, GIF, TIFF, PDF or EMF graphical files, including no scroll bars. Use the Copy method to copy the control's content to the clipboard.

- The **BMP** file format, also known as bitmap image file or device independent bitmap (DIB) file format or simply a bitmap, is a raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter)
- The **JPEG** file format (seen most often with the .jpg extension) is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography.
- The **GIF** (Graphics Interchange Format) is a bitmap image format that was introduced by CompuServe in 1987 and has since come into widespread usage on the World Wide Web due to its wide support and portability.
- The **TIFF** (Tagged Image File Format) is a computer file format for storing raster graphics images, popular among graphic artists, the publishing industry, and both amateur and professional photographers in general.
- The **PNG** (Portable Network Graphics) is a raster graphics file format that supports lossless data compression. PNG was created as an improved, non-patented replacement for Graphics Interchange Format (GIF), and is the most used lossless image compression format on the Internet
- The **PDF** (Portable Document Format) is a file format used to present documents in a manner independent of application software, hardware, and operating systems. Each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, fonts, graphics, and other information needed to display it.
- The EMF (Enhanced Metafile Format) is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following

  Built-in scaling information
  Built-in descriptions that are saved with the file
  Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The following VB sample saves the control's content to a file:

```
If (ChartView1.CopyTo("c:\temp\test.emf")) Then
   MsgBox "test.emf file created, open it using the mspaint editor."
End If
```

The following VB sample prints the EMF content (as bytes, File parameter is empty string):

```
Dim i As Variant
For Each i In ChartView1.CopyTo("")
   Debug.Print i
Next
```

property ChartView.Cursor(Area as ClientAreaEnum) as Variant

Gets or sets the cursor that is displayed when the mouse pointer hovers the control.

Type	Description
Area as ClientAreaEnum	A ClientAreaEnum expression that indicates the control's zone where the mouse pointer is changed.
Variant	A string expression that indicates a predefined value listed bellow, a string expression that indicates the path to a cursor file, a long expression that indicates the handle of the cursor.

Use the Cursor property to specify the cursor that control displays when mouse pointer hovers the parts of the control.

Here's the list of predefined values (string expressions):

- **"exDefault"** - (Default) Shape determined by the object.
- **"exArrow"** - Arrow.
- **"exCross"** - Cross (cross-hair pointer).
- **"exIBeam"** - I Beam.
- **"exIcon"** - Icon (small square within a square).
- **"exSize" -** Size (four-pointed arrow pointing north, south, east, and west).
- **"exSizeNESW"** - Size NE SW (double arrow pointing northeast and southwest).
- **"exSizeNS"** - Size N S (double arrow pointing north and south).
- **"exSizeNWSE"** - Size NW, SE.
- **"exSizeWE"** - Size W E (double arrow pointing west and east).
- **"exUpArrow"** - Up Arrow.
- **"exHourglass"** - Hourglass (wait).
- **"exNoDrop"** - No Drop.
- **"exArrowHourglass"** - Arrow and hourglass.
- **"exHelp"** - Arrow and question mark.
- **"exSizeAll"** - Size all
- **"exHand"** - Hand cursor.

If the cursor value is a string expression, the control looks first if it is not a predefined value like listed above, and if not, it tries to load the cursor from a file. If the Cursor property is a long expression it always indicates a handle to a cursor. The API functions like: LoadCursor or LoadCursorFromFile retrieves a handle to a cursor. In .NET framework, the Handle parameter of the Cursor object specifies the handle to the cursor. Use the Cursors object to access to the list of predefined cursors in the .NET framework.

The following VB sample changes the cursor while the mouse pointer hovers the control's

line number bar:

```
With ChartView1
   .Cursor(exChartArea) = "exCross"
End With
```

Here's the VB.NET alternative:

```
AxChartView1.Ctlset_Cursor(EXCHARTVIEWLib.ClientAreaEnum.exChartArea,
"exCross")
```

The following sample loads a cursor from a file:

```
With ChartView1
   .Cursor(exChartArea) = "C:\WINNT\Cursors\metronom.ani"
End With
```

And here's the VB.NET alternative:

```
AxChartView1.Ctlset_Cursor(EXCHARTVIEWLib.ClientAreaEnum.exChartArea,
"C:\WINNT\Cursors\metronom.ani")
```

The following VB.NET sample changes the cursor with one that Cursors object defines (PanEast cursor):

```
AxChartView1.Ctlset_Cursor(EXCHARTVIEWLib.ClientAreaEnum.exChartArea,
Cursors.PanEast.Handle)
```

The following C++ sample changes the cursor for the control's client area:

```
m_chartview.SetCursor( 0 /*exChartArea*/, COleVariant( "exHelp" ) );
```

The following C# sample changes the cursor for the control's client area:

```
axChartView1.Ctlset_Cursor(EXORGCHARTLib.ClientAreaEnum.exChartArea, "exHelp");
```

The following VFP sample changes the cursor for the control's client area:

```
with thisform.ChartView1
   .Cursor(0) = "exHelp"
endwith
```

property ChartView.DefaultNodePadding(Edge as PaddingEdgeEnum) as Long

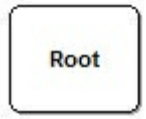
Returns or sets a value that indicates the padding of the nodes in the control.

Type	Description
Edge as PaddingEdgeEnum	A PaddingEdgeEnum expression that specifies the edge to be updated / requested
Long	A long expression that defines the node's default padding

By default, the DefaultNodePadding property is 0, which indicates no padding for any node. The DefaultNodePadding property defines the padding for all nodes. Use the Padding property of the Node to define the padding for specified node. The BackColorNode property defines the node's default background color / EBN object. Use the ForeColorNode property to specify the foreground color for all nodes. Use the FixedHeightNode / FixedWidthNode properties to specify fixed size for all nodes in the chart.

The following screen shot defines a node with no padding (by default):

The following screen shot defines a node with padding:



The following samples show how you can define padding for all nodes:

VBA (MS Access, Excell...)

```
With ChartView1
   .BeginUpdate
   .DefaultNodePadding(-1) = 8
   .IndentSiblingY = 30
   .ShowLinksDir = True
   .PenWidthLink = 2
   .LinkColor = RGB(0,0,0)
   .AntiAliasing = True
   With .Nodes
      .Add "L1 A1",,"LA"
      .Add "L1 B1",,"LB"
```

```
        .Add "L2 A1","LA","LA2"
        .Add "L2 B2","LB","LB2"
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate
End With
```

VB6

```
With ChartView1
    .BeginUpdate
    .DefaultNodePadding(exPaddingAll) = 8
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add "L1 A1",,"LA"
        .Add "L1 B1",,"LB"
        .Add "L2 A1","LA","LA2"
        .Add "L2 B2","LB","LB2"
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate
End With
```

VB.NET

```
With Exchartview1
    .BeginUpdate()

.set_DefaultNodePadding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingAl

    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = Color.FromArgb(0,0,0)
```

```
    .AntiAliasing = True
    With .Nodes
        .Add("L1 A1",,"LA")
        .Add("L1 B1",,"LB")
        .Add("L2 A1","LA","LA2")
        .Add("L2 B2","LB","LB2")
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxChartView1
    .BeginUpdate()
    .set_DefaultNodePadding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,8)
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1 A1",,"LA")
        .Add("L1 B1",,"LB")
        .Add("L2 A1","LA","LA2")
        .Add("L2 B2","LB","LB2")
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate()
End With
```

C++

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

```
    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->BeginUpdate();
spChartView1->PutDefaultNodePadding(EXORGCHARTLib::exPaddingAll,8);
spChartView1->PutIndentSiblingY(30);
spChartView1->PutShowLinksDir(VARIANT_TRUE);
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(RGB(0,0,0));
spChartView1->PutAntiAliasing(VARIANT_TRUE);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
    var_Nodes->Add(L"L1 A1",vtMissing,"LA",vtMissing,vtMissing);
    var_Nodes->Add(L"L1 B1",vtMissing,"LB",vtMissing,vtMissing);
    var_Nodes->Add(L"L2 A1","LA","LA2",vtMissing,vtMissing);
    var_Nodes->Add(L"L2 B2","LB","LB2",vtMissing,vtMissing);
spChartView1->GetNodes()->GetItem("root")->PutCaption(L"Ls As");
spChartView1->EndUpdate();
```

C++ Builder

```
ChartView1->BeginUpdate();
ChartView1-
>DefaultNodePadding[Exorgchartlib_tlb::PaddingEdgeEnum::exPaddingAll] = 8;
ChartView1->IndentSiblingY = 30;
ChartView1->ShowLinksDir = true;
ChartView1->PenWidthLink = 2;
ChartView1->LinkColor = RGB(0,0,0);
ChartView1->AntiAliasing = true;
Exorgchartlib_tlb::INodesPtr var_Nodes = ChartView1->Nodes;
    var_Nodes->Add(L"L1 A1",TNoParam(),TVariant("LA"),TNoParam(),TNoParam());
    var_Nodes->Add(L"L1 B1",TNoParam(),TVariant("LB"),TNoParam(),TNoParam());
    var_Nodes->Add(L"L2 A1",TVariant("LA"),TVariant("LA2"),TNoParam(),TNoParam());
    var_Nodes->Add(L"L2 B2",TVariant("LB"),TVariant("LB2"),TNoParam(),TNoParam());
ChartView1->Nodes->get_Item(TVariant("root"))->Caption = L"Ls As";
```

```
ChartView1->EndUpdate();
```

C#

```
exchartview1.BeginUpdate();
exchartview1.set_DefaultNodePadding(exontrol.EXORGCHARTLib.PaddingEdgeEnum

exchartview1.IndentSiblingY = 30;
exchartview1.ShowLinksDir = true;
exchartview1.PenWidthLink = 2;
exchartview1.LinkColor = Color.FromArgb(0,0,0);
exchartview1.AntiAliasing = true;
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
   var_Nodes.Add("L1 A1",null,"LA",null,null);
   var_Nodes.Add("L1 B1",null,"LB",null,null);
   var_Nodes.Add("L2 A1","LA","LA2",null,null);
   var_Nodes.Add("L2 B2","LB","LB2",null,null);
exchartview1.Nodes["root"].Caption = "Ls As";
exchartview1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   ChartView1.BeginUpdate();
   ChartView1.DefaultNodePadding(-1) = 8;
   ChartView1.IndentSiblingY = 30;
   ChartView1.ShowLinksDir = true;
   ChartView1.PenWidthLink = 2;
   ChartView1.LinkColor = 0;
   ChartView1.AntiAliasing = true;
```

```
    var var_Nodes = ChartView1.Nodes;
        var_Nodes.Add("L1 A1",null,"LA",null,null);
        var_Nodes.Add("L1 B1",null,"LB",null,null);
        var_Nodes.Add("L2 A1","LA","LA2",null,null);
        var_Nodes.Add("L2 B2","LB","LB2",null,null);
    ChartView1.Nodes.Item("root").Caption = "Ls As";
    ChartView1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With ChartView1
        .BeginUpdate
        .DefaultNodePadding(-1) = 8
        .IndentSiblingY = 30
        .ShowLinksDir = True
        .PenWidthLink = 2
        .LinkColor = RGB(0,0,0)
        .AntiAliasing = True
        With .Nodes
            .Add "L1 A1",,"LA"
            .Add "L1 B1",,"LB"
            .Add "L2 A1","LA","LA2"
            .Add "L2 B2","LB","LB2"
        End With
        .Nodes.Item("root").Caption = "Ls As"
        .EndUpdate
    End With
```

```
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axChartView1.BeginUpdate();
axChartView1.set_DefaultNodePadding(EXORGCHARTLib.PaddingEdgeEnum.exPaddi

axChartView1.IndentSiblingY = 30;
axChartView1.ShowLinksDir = true;
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
axChartView1.AntiAliasing = true;
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   var_Nodes.Add("L1 A1",null,"LA",null,null);
   var_Nodes.Add("L1 B1",null,"LB",null,null);
   var_Nodes.Add("L2 A1","LA","LA2",null,null);
   var_Nodes.Add("L2 B2","LB","LB2",null,null);
axChartView1.Nodes["root"].Caption = "Ls As";
axChartView1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Node,com_Nodes;
   anytype var_Node,var_Nodes;
   ;

   super();

   exchartview1.BeginUpdate();
   exchartview1.DefaultNodePadding(-1/*exPaddingAll*/,8);
   exchartview1.IndentSiblingY(30);
   exchartview1.ShowLinksDir(true);
```

```
    exchartview1.PenWidthLink(2);
    exchartview1.LinkColor(WinApi::RGB2int(0,0,0));
    exchartview1.AntiAliasing(true);
    var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
        com_Nodes.Add("L1 A1",,"LA");
        com_Nodes.Add("L1 B1",,"LB");
        com_Nodes.Add("L2 A1","LA","LA2");
        com_Nodes.Add("L2 B2","LB","LB2");
    var_Node = COM::createFromObject(exchartview1.Nodes()).Item("root");
com_Node = var_Node;
    com_Node.Caption("Ls As");
    exchartview1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
    BeginUpdate();
    set_DefaultNodePadding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,8);
    IndentSiblingY := 30;
    ShowLinksDir := True;
    PenWidthLink := 2;
    LinkColor := Color.FromArgb(0,0,0);
    AntiAliasing := True;
    with Nodes do
    begin
        Add('L1 A1',Nil,'LA',Nil,Nil);
        Add('L1 B1',Nil,'LB',Nil,Nil);
        Add('L2 A1','LA','LA2',Nil,Nil);
        Add('L2 B2','LB','LB2',Nil,Nil);
    end;
    Nodes.Item['root'].Caption := 'Ls As';
    EndUpdate();
end
```

Delphi (standard)

```
with ChartView1 do
begin
  BeginUpdate();
  DefaultNodePadding[EXORGCHARTLib_TLB.exPaddingAll] := 8;
  IndentSiblingY := 30;
  ShowLinksDir := True;
  PenWidthLink := 2;
  LinkColor := RGB(0,0,0);
  AntiAliasing := True;
  with Nodes do
  begin
    Add('L1 A1',Null,'LA',Null,Null);
    Add('L1 B1',Null,'LB',Null,Null);
    Add('L2 A1','LA','LA2',Null,Null);
    Add('L2 B2','LB','LB2',Null,Null);
  end;
  Nodes.Item['root'].Caption := 'Ls As';
  EndUpdate();
end
```

VFP

```
with thisform.ChartView1
  .BeginUpdate
  .Object.DefaultNodePadding(-1) = 8
  .IndentSiblingY = 30
  .ShowLinksDir = .T.
  .PenWidthLink = 2
  .LinkColor = RGB(0,0,0)
  .AntiAliasing = .T.
  with .Nodes
    .Add("L1 A1",Null,"LA")
    .Add("L1 B1",Null,"LB")
    .Add("L2 A1","LA","LA2")
    .Add("L2 B2","LB","LB2")
  endwith
  .Nodes.Item("root").Caption = "Ls As"
```

```
   .EndUpdate
endwith
```

dBASE Plus

```
local oChartView,var_Nodes

oChartView = form.EXORGCHARTACTIVEXCONTROL1.nativeObject
oChartView.BeginUpdate()
oChartView.Template = [DefaultNodePadding(-1) = 8] //
oChartView.DefaultNodePadding(-1) = 8
oChartView.IndentSiblingY = 30
oChartView.ShowLinksDir = true
oChartView.PenWidthLink = 2
oChartView.LinkColor = 0x0
oChartView.AntiAliasing = true
var_Nodes = oChartView.Nodes
   var_Nodes.Add("L1 A1",null,"LA")
   var_Nodes.Add("L1 B1",null,"LB")
   var_Nodes.Add("L2 A1","LA","LA2")
   var_Nodes.Add("L2 B2","LB","LB2")
oChartView.Nodes.Item("root").Caption = "Ls As"
oChartView.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oChartView as P
Dim var_Nodes as P

oChartView = topparent:CONTROL_ACTIVEX1.activex
oChartView.BeginUpdate()
oChartView.Template = "DefaultNodePadding(-1) = 8" //
oChartView.DefaultNodePadding(-1) = 8
oChartView.IndentSiblingY = 30
oChartView.ShowLinksDir = .t.
oChartView.PenWidthLink = 2
oChartView.LinkColor = 0
```

```
oChartView.AntiAliasing = .t.
var_Nodes = oChartView.Nodes
   var_Nodes.Add("L1 A1",,"LA")
   var_Nodes.Add("L1 B1",,"LB")
   var_Nodes.Add("L2 A1","LA","LA2")
   var_Nodes.Add("L2 B2","LB","LB2")
oChartView.Nodes.Item("root").Caption = "Ls As"
oChartView.EndUpdate()
```

Visual Objects

```
local var_Nodes as INodes

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:[DefaultNodePadding,exPaddingAll] := 8
oDCOCX_Exontrol1:IndentSiblingY := 30
oDCOCX_Exontrol1:ShowLinksDir := true
oDCOCX_Exontrol1:PenWidthLink := 2
oDCOCX_Exontrol1:LinkColor := RGB(0,0,0)
oDCOCX_Exontrol1:AntiAliasing := true
var_Nodes := oDCOCX_Exontrol1:Nodes
   var_Nodes:Add("L1 A1",nil,"LA",nil,nil)
   var_Nodes:Add("L1 B1",nil,"LB",nil,nil)
   var_Nodes:Add("L2 A1","LA","LA2",nil,nil)
   var_Nodes:Add("L2 B2","LB","LB2",nil,nil)
oDCOCX_Exontrol1:Nodes:[Item,"root"]:Caption := "Ls As"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oChartView,var_Nodes

oChartView = ole_1.Object
oChartView.BeginUpdate()
oChartView.DefaultNodePadding(-1,8)
oChartView.IndentSiblingY = 30
```

```
oChartView.ShowLinksDir = true
oChartView.PenWidthLink = 2
oChartView.LinkColor = RGB(0,0,0)
oChartView.AntiAliasing = true
var_Nodes = oChartView.Nodes
   var_Nodes.Add("L1 A1",,"LA")
   var_Nodes.Add("L1 B1",,"LB")
   var_Nodes.Add("L2 A1","LA","LA2")
   var_Nodes.Add("L2 B2","LB","LB2")
oChartView.Nodes.Item("root").Caption = "Ls As"
oChartView.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComDefaultNodePadding OLEexPaddingAll to 8
   Set ComIndentSiblingY to 30
   Set ComShowLinksDir to True
   Set ComPenWidthLink to 2
   Set ComLinkColor to (RGB(0,0,0))
   Set ComAntiAliasing to True
   Variant voNodes
   Get ComNodes to voNodes
   Handle hoNodes
   Get Create (RefClass(cComNodes)) to hoNodes
   Set pvComObject of hoNodes to voNodes
      Get ComAdd of hoNodes "L1 A1" "LA" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "L1 B1" "LB" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "L2 A1" "LA" "LA2" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "L2 B2" "LB" "LB2" Nothing Nothing to Nothing
   Send Destroy to hoNodes
   Variant voNodes1
   Get ComNodes to voNodes1
   Handle hoNodes1
```

```
  Get Create (RefClass(cComNodes)) to hoNodes1
  Set pvComObject of hoNodes1 to voNodes1
    Variant voNode
    Get ComItem of hoNodes1 "root" to voNode
    Handle hoNode
    Get Create (RefClass(cComNode)) to hoNode
    Set pvComObject of hoNode to voNode
      Set ComCaption of hoNode to "Ls As"
    Send Destroy to hoNode
  Send Destroy to hoNodes1
  Send ComEndUpdate
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oChartView
   LOCAL oNodes

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

   oChartView := XbpActiveXControl():new( oForm:drawingArea )
   oChartView:CLSID  := "Exontrol.ChartView.1" /*{F4DFE455-01FE-420E-A088-
64346DCC3791}*/
   oChartView:create(,, {10,60},{610,370} )

      oChartView:BeginUpdate()
      oChartView:SetProperty("DefaultNodePadding",-1/*exPaddingAll*/,8)
      oChartView:IndentSiblingY := 30
```

```
      oChartView:ShowLinksDir := .T.
      oChartView:PenWidthLink := 2
      oChartView:SetProperty("LinkColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oChartView:AntiAliasing := .T.
      oNodes := oChartView:Nodes()
         oNodes:Add("L1 A1",,"LA")
         oNodes:Add("L1 B1",,"LB")
         oNodes:Add("L2 A1","LA","LA2")
         oNodes:Add("L2 B2","LB","LB2")
      oChartView:Nodes:Item("root"):Caption := "Ls As"
      oChartView:EndUpdate()

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

property ChartView.DragOutsideDef as String

Indicates the options to scroll the control's content like speed, step, and so on while user moves a node by drag and drop outside of the control's content.

Type	Description
String	A String expression that specifies the parameters to specify options to scroll the control's content while the user moves a node by drag and drop outside of the control's client area. The string indicates the following parameters, separated by comma: the step (the number of pixels to scroll at once), 16 pixels, by default), the number of speeds to scroll (5 speeds, by default), the distance to increase or decrease the speed to scroll the control's content (16 pixels, by default), the number of mili-seconds to wait until next scroll is performed (64 ms, by default).

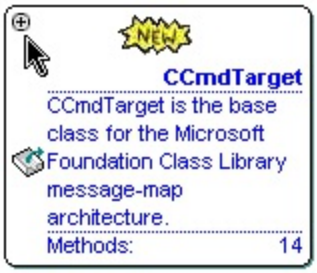
By default, the DragOutsideDef property is "16,5,16,64". The DragOutsideDef property has effect while the AllowMoveNode property is not zero (exDisallow). Use the DragOutsideDef property to customize the speed to scroll the control's content while the user moves a node by drag and drop outside of the control's chart area. The user can scroll the control's content during drag and drop (moving a node), while the cursor hovers outside area of the control. While the cursor is outside of the control and close to the control's margin the speed to scroll is slow. As far as the cursor moves outside far away of control margins the speed is faster.

property ChartView.DrawRoundNode as Boolean

Specifies a value that indicates whether the node has borders with round corners.

Type	Description
Boolean	A boolean expression that indicates whether the nodes have borders with round corners.

Use the DrawRoundNode property to specify whether the nodes have borders with round corners. Use the DrawRoundNode property to specify whether a specified node has borders with round corners. The ShadowNode property determines whether the control displays a shadow for nodes. Use the ShadowNode property to hide the shadow for a specific node. The DrawRoundNode property and ShadowNode property has effect only if no skin is applied to a node. Use the Background property to specify a background color or a visual appearance for specific parts in the control.



method ChartView.EditNode (Node as Variant)

Edits the specified node.

Type	Description
Node as Variant	A Node object to be edited.

Use the EditNode to programmatically edit the giving node. The EditNode method focuses the control's window, ensures that the giving node fits the control's edit node, and edit the node's caption. The EditNode method starts editing the giving node, if the control is enabled and visible, and the node is editable (Editable property). The LayoutStartChanging(exEditNode) event notifies your application once a node is being edited. The LayoutEndChanging(exEditNode) event notifies your application once a node is edited. The Caption property indicates the caption of the node being edited.

property ChartView.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 enable or disable the control. Use the Enabled property to disables a specified node. Use the BackColor property to specify the control's background color. Use the BackColorNode property to define the default background color for nodes. Use the ForeColor property to specify the control's foreground color. Use the ForeColorNode property to define the default foreground color for nodes.

method ChartView.EndUpdate ()

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

Type	Description

The BeginUpdate and EndUpdate methods maintains performance when items are added to the control one at a time. Use the Add method to add new child nodes. Use the Remove method to remove a node from the control. Use the Root property to get the root node of the control.

The following VB sample adds four nodes to the control :

```
With ChartView1
   .BeginUpdate
   With .Nodes
      .Add "Item 1", "root", "Key1"
      .Add "Item 2", "root"
      .Add "Sub Item 1", "Key1"
      .Add "Sub Item 2", "Key1"
   End With
   .EndUpdate
End With
```

The following C++ sample adds four nodes to the control :

```
#include "nodes.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
m_chartview.BeginUpdate();
CNodes nodes = m_chartview.GetNodes();
nodes.Add( "Item 1", COleVariant( "root" ), COleVariant("Key1"), vtMissing, vtMissing );
nodes.Add( "Item 2", COleVariant( "root" ), vtMissing, vtMissing, vtMissing );
nodes.Add( "Sub Item 1", COleVariant( "Key1" ), vtMissing, vtMissing, vtMissing );
nodes.Add( "Sub Item 2", COleVariant( "Key1" ), vtMissing, vtMissing, vtMissing );
m_chartview.EndUpdate();
```

The following VB.NET sample adds four nodes to the control :

```
With AxChartView1
   .BeginUpdate()
```

```
    With .Nodes
        .Add("Item 1", "root", "Key1")
        .Add("Item 2", "root")
        .Add("Sub Item 1", "Key1")
        .Add("Sub Item 2", "Key1")
    End With
    .EndUpdate()
End With
```

The following C# sample adds four nodes to the control :

```
axChartView1.BeginUpdate();
EXORGCHARTLib.Nodes nodes = axChartView1.Nodes;
nodes.Add("Item 1", "root", "Key1", null, null);
nodes.Add("Item 2", "root", null, null, null);
nodes.Add("Sub Item 1", "Key1", null, null, null);
nodes.Add("Sub Item 2", "Key1", null, null, null);
axChartView1.EndUpdate();
```

The following VFP sample adds four nodes to the control :

```
With thisform.ChartView1
    .BeginUpdate
    With .Nodes
        .Add("Item 1", "root", "Key1")
        .Add("Item 2", "root")
        .Add("Sub Item 1", "Key1")
        .Add("Sub Item 2", "Key1")
    EndWith
    .EndUpdate
EndWith
```

method ChartView.EnsureVisibleNode (Node as Variant)

Ensures the given node is in the visible client area.

Type	Description
Node as Variant	A Node object to ensure that it fits the control's client area.

Use the EnsureVisibleNode method to ensure that a node fits the control's client area. Use the SelectNode property to select a node. The ScrollOnEnsure property specifies a value that indicates whether the control scrolls the control's content when ensuring that a node is visible. The control automatically scrolls the control's content to ensure that the node being clicked fits the control's client area, if the EnsureVisibleOnSelect property is True.

The following VB ensures that the node is in the client are when the cursor hovers the node:

```
Private Sub ChartView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .NodeFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (n Is Nothing) Then
         .EnsureVisibleNode n
      End If
   End With
End Sub
```

The following C++ ensures that the node is in the client are when the cursor hovers the node:

```
void OnMouseMoveChartview1(short Button, short Shift, long X, long Y)
{
   CNode node = m_chartview.GetNodeFromPoint( X, Y );
   if ( node.m_lpDispatch != NULL )
      m_chartview.EnsureVisibleNode( COleVariant( node.GetKey() ) );
}
```

The following VB.NET ensures that the node is in the client are when the cursor hovers the node:

```
Private Sub AxChartView1_MouseMoveEvent(ByVal sender As Object, ByVal e As
```

```
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent) Handles
AxChartView1.MouseMoveEvent
    With AxChartView1
        Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(e.x, e.y)
        If Not (n Is Nothing) Then
            .EnsureVisibleNode(n)
        End If
    End With
End Sub
```

The following C# ensures that the node is in the client are when the cursor hovers the node:

```
private void axChartView1_MouseMoveEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent e)
{
    EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(e.x, e.y);
    if (node != null)
        axChartView1.EnsureVisibleNode(node);
}
```

The following VFP ensures that the node is in the client are when the cursor hovers the node:

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

With thisform.ChartView1
    local n
    n = .NodeFromPoint(x , y )
    If !isnull(n) then
        .EnsureVisibleNode(n)
    EndIf
EndWith
```

property ChartView.EnsureVisibleOnSelect as Boolean

Retrieves or sets a value that indicates whether the control ensures the selected node is visible.

Type	Description
Boolean	A boolean expression that indicates whether the control ensures the selected node is visible, when user clicks a node.

Use the EnsureVisibleOnSelect property to let control ensures that selected node is visible, when user clicks the node. Use the EnsureVisibleNode method to programmatically ensure that a specified node is visible. Use the SelectNode property to select a node. The control fires the Select event when a node is selected. The SelColor property retrieves or sets a value that indicates the color used to mark the selected node. The ScrollOnEnsure property specifies a value that indicates whether the control scrolls the control's content when ensuring that a node is visible.

property ChartView.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 ChartView.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 control's background color:

```
Debug.Print ChartView1.ExecuteTemplate("BackColor")
```

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 ChartView.ExpandOnDblClk as Boolean

Expands or collapses a node when the user dbl clicks the node.

Type	Description
Boolean	A boolean expression that indicates whether the control expands or collapses a node when user dbl clicks the node.

By default, the ExpandOnDblClk property is True. Use the HasButtons property to display the +/- buttons for nodes that contain child nodes. Use the HasButton property to hide the +/- button inside a particular node. Use the Expanded property to expand or collapse a node by code. The control fires the Expand event when user expands or collapse a node. Use the ButtonsAlign property to specify the position of +/- buttons inside nodes. The control fires the DblClick event when the user double clicks. Use the NodeFromPoint property to determine the node from the point.

property ChartView.ExploreFromNode as Variant

Explores the organigram from the node.

Type	Description
Variant	A long expression that indicates the index's of the node being explored, a string expression that indicates the key of the node, or a Node object that indicates the reference to the node being explored.

Use the ExploreFromNode property to define the root node being displayed. By default, the ExploreFromHere property points to the Root node of the organigram. Use the Key property to specify the key of the node. Use the Caption property to specify the caption of the node. Use the ExpandOnDblClick property to disable expanding a node when the user double clicks the node. Use the SelectNode property to specify the selected node.

The following VB sample explores the node being double clicked:

```
Private Sub ChartView1_DblClick(Shift As Integer, x As Single, Y As Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .NodeFromPoint(x / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (n Is Nothing) Then
         .ExploreFromNode = n
      End If
   End With
End Sub
```

The following C++ sample explores the node being double clicked:

```
#include "node.h"
void OnDblClickChartview1(short Shift, long X, long Y)
{
   CNode node = m_chartview.GetNodeFromPoint( X, Y );
   if ( node.m_lpDispatch != NULL )
      m_chartview.SetExploreFromNode( COleVariant( node.GetKey() ) );
}
```

The following VB.NET sample explores the node being double clicked:

```
Private Sub AxChartView1_DblClick(ByVal sender As Object, ByVal e As
```

```
AxEXORGCHARTLib._IChartViewEvents_DblClickEvent) Handles AxChartView1.DblClick
    With AxChartView1
        Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(e.x, e.y)
        If Not (n Is Nothing) Then
            .ExploreFromNode = n
        End If
    End With
End Sub
```

The following C# sample explores the node being double clicked:

```
private void axChartView1_DblClick(object sender,
AxEXORGCHARTLib._IChartViewEvents_DblClickEvent e)
{
    EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(e.x, e.y);
    if (node != null)
        axChartView1.ExploreFromNode = node;
}
```

The following VFP sample explores the node being double clicked:

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

With thisform.ChartView1
    local n
    n = .NodeFromPoint(x , y )
    If !isnull(n) then
        .ExploreFromNode = n
    EndIf
EndWith
```

property ChartView.FixedHeightNode as Long

Retrieves or sets a value that indicates whether the height of the node's caption is fixed.

Type	Description
Long	A long expression that indicates the maximum height for the node's caption.

By default, the FixedHeightNode property is -1. Use the FixedHeightNode and FixedWidthNode properties to specify fixed size for all nodes in the organigram. Use the FixedHeight and FixedWidth properties to define the size of a specified node. If the FixedHeighNode is negative the height of the node's caption is automatically computed. Use the Font property to specify the control's font. The DefaultNodePadding property defines the padding for all nodes.

property ChartView.FixedWidthNode as Long

Retrieves or sets a value that indicates whether the width of the node's caption is fixed.

Type	Description
Long	A long expression that defines the width for all node captions in the organigram.

Use the FixedHeightNode and FixedWidthNode properties to specify fixed size for all nodes in the organigram. Use the FixedHeight and FixedWidth properties to define the size of a specified node. By default, the FixedWidthNode property is -1. If the FixedWidthNode is negative the width of the node's caption is automatically computed. Use the Caption property to specify the node's caption. The DefaultNodePadding property defines the padding for all nodes.

property ChartView.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 define the font for all nodes. Use the <b>, <i>, <s>, or <u> HTML tags in the Caption property to define the font attributes being used for a specific node. Use the BeginUpdate and EndUpdate method to maintain performance while adding new columns or items. Use the Refresh method to refresh the control.

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

```
With ChartView1
   With .Font
      .Name = "Tahoma"
   End With
   .Refresh
End With
```

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

```
COleFont font = m_chartview.GetFont();
font.SetName( "Tahoma" );
m_chartview.Refresh();
```

the C++ sample requires definition of COleFont class (#include "Font.h")

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

```
With AxChartView1
   Dim font As System.Drawing.Font = New System.Drawing.Font("Tahoma", 10,
FontStyle.Regular, GraphicsUnit.Point)
   .Font = font
   .CtlRefresh()
End With
```

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

```
System.Drawing.Font font = new System.Drawing.Font("Tahoma", 10, FontStyle.Regular);
```

```
axChartView1.Font = font;
axChartView1.CtlRefresh();
```

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

```
with thisform.ChartView1.Object
   .Font.Name = "Tahoma"
   .Refresh()
endwith
```

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

```
Font
{
   Name = "Tahoma"
}
```

property ChartView.ForeColor as Color

Specifies the control's foreground color.

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

Use the ForeColor property to define the control's foreground color. Use the ForeColorNode property to define the default foreground color for nodes. Use the ForeColor property to define the foreground color for a specified node. Use the Picture property to display a picture on the control's foreground. Use the Font property to specify the control's font.

property ChartView.ForeColorNode as Color

Specifies the default node's foreground color.

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

Use the ForeColorNode property to define the default foreground color for nodes. Use the ForeColor property to define the foreground color for a specific node. Use the ClearForeColor method to clear the node's foreground color. Use the BackColor property to specify the control's background color. The DefaultNodePadding property defines the padding for all nodes.

method ChartView.FormatABC (Expression as String, [A as Variant], [B as Variant], [C as Variant])

Formats the A,B,C values based on the giving expression and returns the result.

Type	Description
Expression as String	A String that defines the expression to be evaluated.
A as Variant	A VARIANT expression that indicates the value of the A keyword.
B as Variant	A VARIANT expression that indicates the value of the B keyword.
C as Variant	A VARIANT expression that indicates the value of the C keyword.
Return	**Description**
Variant	A VARIANT expression that indicates the result of the evaluation the ChartView.

The FormatABC method formats the A,B,C values based on the giving expression and returns the result.

For instance:

- "A + B + C", adds / concatenates the values of the A, B and C
- "value MIN 0 MAX 99", limits the value between 0 and 99
- "value format ``", formats the value with two decimals, according to the control's panel setting
- "date(`now`)" returns the current time as double

The FormatABC method supports the following keywords, constants, operators and functions:

- **A** or **value** keyword, indicates a variable A whose value is giving by the A parameter
- **B** keyword, indicates a variable B whose value is giving by the B parameter
- **C** keyword, indicates a variable C whose value is giving by the C parameter

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

property ChartView.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 ChartView.FrameFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Frame

Gets the frame from point.

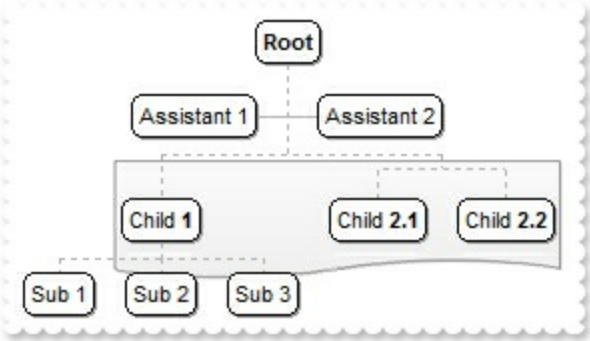
Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates
Frame	A Frame object where the point is.

Use the FrameFromPoint property to determine the frame from specified position. **If the X parameter is -1 and Y parameter is -1 the NodeFromPoint property determines the frame from the cursor**.

A Frame is defined by an union of nodes, and can:

- specify whether the frame is shown on the back or on the front using the ShowOnBackground property
- define the padding of the frame using the Padding property
- define a solid or EBN background color to be displayed on the frame's background, using the BackColor property of the Frame object
- The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background.
- define a different border or pattern to be shown, using the Pattern property

The following screen show shows an EBN frame around child nodes of the root node:



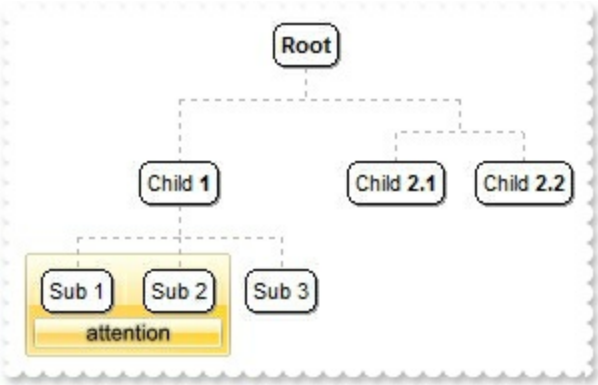
property ChartView.Frames as Frames

Gets the control's collection of frames.

Type	Description
Frames	A Frames collection that holds Frame objects.

The control supports frames, that provide unlimited options to show any HTML text, images, colors, EBNs, patterns, borders, anywhere on the frame's background, that's defined by union of any node on the chart. Use the Add method of the Frames collection to add new frames to the chart. The Padding property defines frame's padding. The Nodes property defines the nodes to be included in the frame.

The following screen shot shows a frame around Sub 1 and Sub 2 nodes.



Is it possible to show some text/caption on the frame's background?

VBA (MS Access, Excell...)

```
With ChartView1
   .BeginUpdate
   .IndentSiblingY = 32
   .VisualAppearance.Add
1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAAzQFAYawdBgABoGUZ4
 & _
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 & _
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 & _
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 & _
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
```

```
 & _
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xISXYQicMF
 & _
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnI
 & _
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 & _
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 & _
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 & _
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 & _
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 & _
"AwAggAIAYFQQ4WgEggDIAQgI="
   With .Nodes
      .Add "Child <b>1</b>",,"1234"
      .Add "Sub 1","1234","AK1"
      .Add "Sub 2","1234","AK2"
      .Add "Sub 3","1234"
      .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
   End With
   With .Frames.Add("AK1,AK2")
      .Padding(-1) = 8
      .Padding(3) = 22
      .BackColor = &H1000000
      .Pattern.Type = 0
      .BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
   End With
   .EndUpdate
End With
```

VB6

```
With ChartView1
   .BeginUpdate
   .IndentSiblingY = 32
   .VisualAppearance.Add
1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAAzQFAYawdBgABoGUZ4
 & _
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 & _
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 & _
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 & _
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 & _
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 & _
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnl
 & _
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 & _
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 & _
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 & _
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 & _
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 & _
"AwAggAIAYFQQ4WgEggDIAQgI="
   With .Nodes
      .Add "Child <b>1</b>",,"1234"
      .Add "Sub 1","1234","AK1"
      .Add "Sub 2","1234","AK2"
      .Add "Sub 3","1234"
      .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
   End With
   With .Frames.Add("AK1,AK2")
```

```
        .Padding(exPaddingAll) = 8
        .Padding(exPaddingBottom) = 22
        .BackColor = &H1000000
        .Pattern.Type = exPatternEmpty
        .BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
    End With
    .EndUpdate
End With
```

VB.NET

```
With Exchartview1
    .BeginUpdate()
    .IndentSiblingY = 32

.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKA
 & _
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 & _
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 & _
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 & _
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 & _
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 & _
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnl
 & _
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 & _
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 & _
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 & _
```

```
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 & _
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 & _
"AwAggAIAYFQQ4WgEggDIAQgI=")
   With .Nodes
      .Add("Child <b>1</b>",,"1234")
      .Add("Sub 1","1234","AK1")
      .Add("Sub 2","1234","AK2")
      .Add("Sub 3","1234")
      .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
   End With
   With .Frames.Add("AK1,AK2")
      .set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,8)
      .set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingBottom,22)
      .BackColor32 = &H1000000
      .Pattern.Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternEmpty
      .BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
   End With
   .EndUpdate()
End With
```

VB.NET for /COM

```
With AxChartView1
   .BeginUpdate()
   .IndentSiblingY = 32

.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKA
 & _
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 & _
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 & _
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
```

```
 & _
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 & _
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 & _
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnI
 & _
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 & _
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 & _
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 & _
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 & _
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 & _
"AwAggAIAYFQQ4WgEggDIAQgI=")
   With .Nodes
      .Add("Child <b>1</b>",,"1234")
      .Add("Sub 1","1234","AK1")
      .Add("Sub 2","1234","AK2")
      .Add("Sub 3","1234")
      .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
   End With
   With .Frames.Add("AK1,AK2")
      .Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll) = 8
      .Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingBottom) = 22
      .BackColor = &H1000000
      .Pattern.Type = EXORGCHARTLib.PatternEnum.exPatternEmpty
      .BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
   End With
   .EndUpdate()
End With
```

C++

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

    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->BeginUpdate();
spChartView1->PutIndentSiblingY(32);
spChartView1->GetVisualAppearance()-
>Add(1,_bstr_t("gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAAzQFAYaw
 +
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 +
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 +
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 +
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThIk4N
 +
"mkUJ/B8JBBkIRAYmQcRglwP5lAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 +
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnI
 +
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 +
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 +
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 +
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 +
```

```
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 +
"AwAggAIAYFQQ4WgEggDIAQgI=");
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"Child <b>1</b>",vtMissing,"1234",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 1","1234","AK1",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 2","1234","AK2",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 3","1234",vtMissing,vtMissing,vtMissing);
   var_Nodes->Add(L"Child <b>2.1</b>",vtMissing,vtMissing,vtMissing,vtMissing)-
>AddGroup(L"Child <b>2.2</b>",vtMissing,vtMissing);
EXORGCHARTLib::IFramePtr var_Frame = spChartView1->GetFrames()-
>Add("AK1,AK2");
   var_Frame->PutPadding(EXORGCHARTLib::exPaddingAll,8);
   var_Frame->PutPadding(EXORGCHARTLib::exPaddingBottom,22);
   var_Frame->PutBackColor(0x1000000);
   var_Frame->GetPattern()->PutType(EXORGCHARTLib::exPatternEmpty);
   var_Frame-
>PutBackgroundExt(L"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x0100
 ;;0>attention`,align=0x11]");
spChartView1->EndUpdate();
```

C++ Builder

```
ChartView1->BeginUpdate();
ChartView1->IndentSiblingY = 32;
ChartView1->VisualAppearance-
>Add(1,TVariant(String("gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAA
 +
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 +
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 +
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 +
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThIk4N
 +
```

```
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 +
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnl
 +
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 +
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 +
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 +
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 +
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 +
"AwAggAIAYFQQ4WgEggDIAQgI="));
Exorgchartlib_tlb::INodesPtr var_Nodes = ChartView1->Nodes;
   var_Nodes->Add(L"Child
<b>1</b>",TNoParam(),TVariant("1234"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub
1",TVariant("1234"),TVariant("AK1"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub
2",TVariant("1234"),TVariant("AK2"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub 3",TVariant("1234"),TNoParam(),TNoParam(),TNoParam());
   var_Nodes->Add(L"Child
<b>2.1</b>",TNoParam(),TNoParam(),TNoParam(),TNoParam())->AddGroup(L"Child
<b>2.2</b>",TNoParam(),TNoParam());
Exorgchartlib_tlb::IFramePtr var_Frame = ChartView1->Frames-
>Add(TVariant("AK1,AK2"));
   var_Frame->set_Padding(Exorgchartlib_tlb::PaddingEdgeEnum::exPaddingAll,8);
   var_Frame-
>set_Padding(Exorgchartlib_tlb::PaddingEdgeEnum::exPaddingBottom,22);
   var_Frame->BackColor = 0x1000000;
   var_Frame->Pattern->Type = Exorgchartlib_tlb::PatternEnum::exPatternEmpty;
   var_Frame->BackgroundExt =
L"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]";
ChartView1->EndUpdate();
```

C#

```
exchartview1.BeginUpdate();
exchartview1.IndentSiblingY = 32;
exchartview1.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxl
 +
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 +
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 +
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 +
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 +
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 +
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnl
 +
"YOxSIIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 +
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 +
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 +
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 +
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 +
"AwAggAIAYFQQ4WgEggDIAQgI=");
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
   var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
   var_Nodes.Add("Sub 1","1234","AK1",null,null);
   var_Nodes.Add("Sub 2","1234","AK2",null,null);
   var_Nodes.Add("Sub 3","1234",null,null,null);
   var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
```

```
<b>2.2</b>",null,null);
exontrol.EXORGCHARTLib.Frame var_Frame = exchartview1.Frames.Add("AK1,AK2");

var_Frame.set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,8);

var_Frame.set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingBottom

   var_Frame.BackColor32 = 0x1000000;
   var_Frame.Pattern.Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternEmpty;
   var_Frame.BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]";
exchartview1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   ChartView1.BeginUpdate();
   ChartView1.IndentSiblingY = 32;

ChartView1.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRD
 +

"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 +

"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 +

"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
```

```
 +
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 +
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 +
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnl
 +
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 +
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 +
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 +
"O/sVgQRkv7EyO4P4mgeC5E+MsTIvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 +
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 +
  "AwAggAIAYFQQ4WgEggDIAQgI=");
  var var_Nodes = ChartView1.Nodes;
     var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
     var_Nodes.Add("Sub 1","1234","AK1",null,null);
     var_Nodes.Add("Sub 2","1234","AK2",null,null);
     var_Nodes.Add("Sub 3","1234",null,null,null);
     var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
  var var_Frame = ChartView1.Frames.Add("AK1,AK2");
     var_Frame.Padding(-1) = 8;
     var_Frame.Padding(3) = 22;
     var_Frame.BackColor = 16777216;
```

```
        var_Frame.Pattern.Type = 0;
        var_Frame.BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]";
    ChartView1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With ChartView1
        .BeginUpdate
        .IndentSiblingY = 32
        .VisualAppearance.Add
1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAAzQFAYawdBgABoGUZ4
 & _

"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 & _

"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 & _

"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 & _

"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 & _
```

```
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 & _

"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnl
 & _

"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 & _

"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 & _

"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 & _

"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 & _

"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 & _
  "AwAggAIAYFQQ4WgEggDIAQgI="
    With .Nodes
      .Add "Child <b>1</b>",,"1234"
      .Add "Sub 1","1234","AK1"
      .Add "Sub 2","1234","AK2"
      .Add "Sub 3","1234"
      .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
    End With
    With .Frames.Add("AK1,AK2")
      .Padding(-1) = 8
      .Padding(3) = 22
      .BackColor = &H1000000
      .Pattern.Type = 0
      .BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
    End With
```

```
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axChartView1.BeginUpdate();
axChartView1.IndentSiblingY = 32;
axChartView1.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAx
 +
"IMIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 +
"XpCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUK
 +
"RjqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgT
 +
"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 +
"mkUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMF
 +
"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnl
 +
"YOxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOS
 +
"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 +
"SQsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8
 +
"O/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5
 +
"fjjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
 +
"AwAggAIAYFQQ4WgEggDIAQgI=");
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
```

```
   var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
   var_Nodes.Add("Sub 1","1234","AK1",null,null);
   var_Nodes.Add("Sub 2","1234","AK2",null,null);
   var_Nodes.Add("Sub 3","1234",null,null,null);
   var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
EXORGCHARTLib.Frame var_Frame = axChartView1.Frames.Add("AK1,AK2");
   var_Frame.set_Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,8);
   var_Frame.set_Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingBottom,22);
   var_Frame.BackColor = 0x1000000;
   var_Frame.Pattern.Type = EXORGCHARTLib.PatternEnum.exPatternEmpty;
   var_Frame.BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]";
axChartView1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Frame,com_Node,com_Nodes,com_Pattern;
   anytype var_Frame,var_Node,var_Nodes,var_Pattern;
   str var_s;
   ;

   super();

   exchartview1.BeginUpdate();
   exchartview1.IndentSiblingY(32);
   var_s =
"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAAzQFAYawdBgABoGUZ4J

   var_s = var_s +
"MIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS

   var_s = var_s +
```

```
"pCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUKcV

  var_s = var_s +
"jqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgTi

  var_s = var_s +
"IKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N

  var_s = var_s +
"kUJ/B8JBBkIRAYmQcRglwP5lAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMR0

  var_s = var_s +
"oTufwwkeloskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnlP

  var_s = var_s +
"OxSIIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLI7GANRuiuMBajsBQLEgRY0gOSx

  var_s = var_s +
"DiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY58

  var_s = var_s +
"QsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8j

  var_s = var_s +
"/sVgQRkv7EyO4P4mgeC5E+MsTIvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5F

  var_s = var_s +
"jjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom

  var_s = var_s + "wAggAIAYFQQ4WgEggDIAQgI=";
  exchartview1.VisualAppearance().Add(1,COMVariant::createFromStr(var_s));
  var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
    com_Nodes.Add("Child <b>1</b>",,"1234");
    com_Nodes.Add("Sub 1","1234","AK1");
    com_Nodes.Add("Sub 2","1234","AK2");
    com_Nodes.Add("Sub 3","1234");
    var_Node = COM::createFromObject(com_Nodes.Add("Child <b>2.1</b>"));
```

```
com_Node = var_Node;
        com_Node.AddGroup("Child <b>2.2</b>");
    var_Frame = COM::createFromObject(exchartview1.Frames()).Add("AK1,AK2");
com_Frame = var_Frame;
        com_Frame.Padding(-1/*exPaddingAll*/,8);
        com_Frame.Padding(3/*exPaddingBottom*/,22);
        com_Frame.BackColor(0x1000000);
        var_Pattern = COM::createFromObject(com_Frame.Pattern()); com_Pattern =
var_Pattern;
        com_Pattern.Type(0/*exPatternEmpty*/);

com_Frame.BackgroundExt("top[4],left[4],right[4],client,bottom[4],bottom[16,back=0
 ;;0>attention`,align=0x11]");
    exchartview1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
    BeginUpdate();
    IndentSiblingY := 32;

VisualAppearance.Add(1,'gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKA
 +

'MIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 +

'pCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUKcV
 +

'jqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgTi
 +

'IKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThIk4N
 +
```

```
'kUJ/B8JBBkIRAYmQcRgIwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMR0
 +
'oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnIP
 +
'OxSlIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOSx
 +
'DiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY58
 +
'QsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAyISSg8j
 +
'/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5F
 +
'jjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMomv
 +
  'wAggAIAYFQQ4WgEggDIAQgI=');
  with Nodes do
  begin
    Add('Child <b>1</b>',Nil,'1234',Nil,Nil);
    Add('Sub 1','1234','AK1',Nil,Nil);
    Add('Sub 2','1234','AK2',Nil,Nil);
    Add('Sub 3','1234',Nil,Nil,Nil);
    Add('Child <b>2.1</b>',Nil,Nil,Nil,Nil).AddGroup('Child <b>2.2</b>',Nil,Nil);
  end;
  with Frames.Add('AK1,AK2') do
  begin
    Padding[EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll] := 8;
    Padding[EXORGCHARTLib.PaddingEdgeEnum.exPaddingBottom] := 22;
    BackColor := $1000000;
    Pattern.Type := EXORGCHARTLib.PatternEnum.exPatternEmpty;
    BackgroundExt :=
```

```
'top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]';
   end;
   EndUpdate();
end
```

Delphi (standard)

```
with ChartView1 do
begin
   BeginUpdate();
   IndentSiblingY := 32;

VisualAppearance.Add(1,'gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKA
 +

'MIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS
 +

'pCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUKcV
 +

'jqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgTi
 +

'IKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThIk4N
 +

'kUJ/B8JBBkIRAYmQcRglwP5IAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMR0
 +

'oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnIP
 +

'OxSIIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLl7GANRuiuMBajsBQLEgRY0gOSx
 +
```

```
'DiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY58
 +

'QsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAyISSg8j
 +

'/sVgQRkv7EyO4P4mgeC5E+MsTlvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5F
 +

'jjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMomv
 +
   'wAggAIAYFQQ4WgEggDIAQgI=');
   with Nodes do
   begin
      Add('Child <b>1</b>',Null,'1234',Null,Null);
      Add('Sub 1','1234','AK1',Null,Null);
      Add('Sub 2','1234','AK2',Null,Null);
      Add('Sub 3','1234',Null,Null,Null);
      Add('Child <b>2.1</b>',Null,Null,Null,Null).AddGroup('Child
<b>2.2</b>',Null,Null);
   end;
   with Frames.Add('AK1,AK2') do
   begin
      Padding[EXORGCHARTLib_TLB.exPaddingAll] := 8;
      Padding[EXORGCHARTLib_TLB.exPaddingBottom] := 22;
      BackColor := $1000000;
      Pattern.Type := EXORGCHARTLib_TLB.exPatternEmpty;
      BackgroundExt :=
'top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]';
   end;
   EndUpdate();
end
```

VFP

```
with thisform.ChartView1
```

```
  .BeginUpdate
  .IndentSiblingY = 32
  var_s =
"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAAzQFAYawdBgABoGUZ4J

  var_s = var_s +
"MIRLE6PZgjOYZchqRYZSjAdIzUAFFr1J4AKbfKJpfoKBJERrScgxeBUNAZBKlY7paoKSpS

  var_s = var_s +
"pCMjVMAGGQPIa8MLCSwIJBaHqUQLZNixLStAyxHCtKKhG+vJwHL6BcqnPKKRqSUKcV

  var_s = var_s +
"jqAZpCkeoNC8P4DDiWp3nYVwdkaSgrGGVhSnSHJum6CgGAiBIUA0JgziGVJkGWRgTi

  var_s = var_s +
"IKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N

  var_s = var_s +
"kUJ/B8JBBkIRAYmQcRglwP5lAOfhNDKCAikSRgxE8WJyEQIJkEGdhHhuD5xlSXYQicMR0

  var_s = var_s +
"oTufwwkeloskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZnlP

  var_s = var_s +
"OxSIIFptCERpSBGbQgn6SItmyUwAHaM4MgqXBljSLI7GANRuiuMBajsBQLEgRY0gOSx

  var_s = var_s +
"DiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY58

  var_s = var_s +
"QsDSHQaFeUQIAKRJIDuT5DnyQ4tnWfR7GCLZMBqXI+i2S5KlyOvukOfxZDICAylSSg8j

  var_s = var_s +
"/sVgQRkv7EyO4P4mgeC5E+MsTIvBfgWD+PFBIFgvgaTaPwP4mxdA/CmNQOoWgci5F

  var_s = var_s +
"jjE+B0H4dA1ASKcK8UYnxzhXAyOsPQvhfgYDoPEQ4RwMBziqDYXY6g9AmFyCwMom
```

```
   var_s = var_s + "wAggAIAYFQQ4WgEggDIAQgI="
   .VisualAppearance.Add(1,var_s)
   with .Nodes
      .Add("Child <b>1</b>",Null,"1234")
      .Add("Sub 1","1234","AK1")
      .Add("Sub 2","1234","AK2")
      .Add("Sub 3","1234")
      .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
   endwith
   with .Frames.Add("AK1,AK2")
      .Padding(-1) = 8
      .Padding(3) = 22
      .BackColor = 0x1000000
      .Pattern.Type = 0
      .BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
local oChartView,var_Frame,var_Nodes

oChartView = form.EXORGCHARTACTIVEXCONTROL1.nativeObject
oChartView.BeginUpdate()
oChartView.IndentSiblingY = 32
oChartView.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRD
 ;

+"5oTufwwkeIoskIMYPDOfB0iOaZznwbw2GoKoQGaKQFjqEBQikBBjGCf4nCgShoLyTCZ
 &
 +"AwAggAIAYFQQ4WgEggDIAQgI=")
var_Nodes = oChartView.Nodes
   var_Nodes.Add("Child <b>1</b>",null,"1234")
```

```
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Frame = oChartView.Frames.Add("AK1,AK2")
   // var_Frame.Padding(-1) = 8
   with (oChartView)
      TemplateDef = [dim var_Frame]
      TemplateDef = var_Frame
      Template = [var_Frame.Padding(-1) = 8]
   endwith
   // var_Frame.Padding(3) = 22
   with (oChartView)
      TemplateDef = [dim var_Frame]
      TemplateDef = var_Frame
      Template = [var_Frame.Padding(3) = 22]
   endwith
   var_Frame.BackColor = 0x1000000
   var_Frame.Pattern.Type = 0
   var_Frame.BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
oChartView.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oChartView as P
Dim var_Frame as P
Dim var_Nodes as P

oChartView = topparent:CONTROL_ACTIVEX1.activex
oChartView.BeginUpdate()
oChartView.IndentSiblingY = 32
oChartView.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRD

var_Nodes = oChartView.Nodes
```

```
   var_Nodes.Add("Child <b>1</b>",,"1234")
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Frame = oChartView.Frames.Add("AK1,AK2")
   ' var_Frame.Padding(-1) = 8
   oChartView.TemplateDef = "dim var_Frame"
   oChartView.TemplateDef = var_Frame
   oChartView.Template = "var_Frame.Padding(-1) = 8"

   ' var_Frame.Padding(3) = 22
   oChartView.TemplateDef = "dim var_Frame"
   oChartView.TemplateDef = var_Frame
   oChartView.Template = "var_Frame.Padding(3) = 22"

   var_Frame.BackColor = 16777216
   var_Frame.Pattern.Type = 0
   var_Frame.BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
oChartView.EndUpdate()
```

Visual Objects

```
local var_Frame as IFrame
local var_Nodes as INodes

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:IndentSiblingY := 32
oDCOCX_Exontrol1:VisualAppearance:Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAAD

var_Nodes := oDCOCX_Exontrol1:Nodes
   var_Nodes:Add("Child <b>1</b>",nil,"1234",nil,nil)
   var_Nodes:Add("Sub 1","1234","AK1",nil,nil)
   var_Nodes:Add("Sub 2","1234","AK2",nil,nil)
```

```
   var_Nodes:Add("Sub 3","1234",nil,nil,nil)
   var_Nodes:Add("Child <b>2.1</b>",nil,nil,nil,nil):AddGroup("Child
<b>2.2</b>",nil,nil)
var_Frame := oDCOCX_Exontrol1:Frames:Add("AK1,AK2")
   var_Frame:[Padding,exPaddingAll] := 8
   var_Frame:[Padding,exPaddingBottom] := 22
   var_Frame:BackColor := 0x1000000
   var_Frame:Pattern:Type := exPatternEmpty
   var_Frame:BackgroundExt :=
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oChartView,var_Frame,var_Nodes

oChartView = ole_1.Object
oChartView.BeginUpdate()
oChartView.IndentSiblingY = 32
oChartView.VisualAppearance.Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRD
 &

+"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nK

var_Nodes = oChartView.Nodes
   var_Nodes.Add("Child <b>1</b>",,"1234")
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Frame = oChartView.Frames.Add("AK1,AK2")
   var_Frame.Padding(-1,8)
   var_Frame.Padding(3,22)
   var_Frame.BackColor = 16777216 /*0x1000000*/
   var_Frame.Pattern.Type = 0
```

```
   var_Frame.BackgroundExt =
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
oChartView.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComIndentSiblingY to 32
   Variant voAppearance
   Get ComVisualAppearance to voAppearance
   Handle hoAppearance
   Get Create (RefClass(cComAppearance)) to hoAppearance
   Set pvComObject of hoAppearance to voAppearance
      Get ComAdd of hoAppearance 1
"gBFLBCJwBAEHhEJAADhABWMMACAADACAxRDAMgBQKAAzQFAYawdBgABoGUZ4J
 to Nothing
   Send Destroy to hoAppearance
   Variant voNodes
   Get ComNodes to voNodes
   Handle hoNodes
   Get Create (RefClass(cComNodes)) to hoNodes
   Set pvComObject of hoNodes to voNodes
      Get ComAdd of hoNodes "Child <b>1</b>" "1234" Nothing Nothing to
Nothing
      Get ComAdd of hoNodes "Sub 1" "1234" "AK1" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "Sub 2" "1234" "AK2" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "Sub 3" "1234" Nothing Nothing Nothing to Nothing
      Variant voNode
      Get ComAdd of hoNodes "Child <b>2.1</b>" Nothing Nothing Nothing
Nothing to voNode
      Handle hoNode
      Get Create (RefClass(cComNode)) to hoNode
      Set pvComObject of hoNode to voNode
```

```
        Get ComAddGroup of hoNode "Child <b>2.2</b>" Nothing Nothing to
Nothing
      Send Destroy to hoNode
   Send Destroy to hoNodes
   Variant voFrames
   Get ComFrames to voFrames
   Handle hoFrames
   Get Create (RefClass(cComFrames)) to hoFrames
   Set pvComObject of hoFrames to voFrames
      Variant voFrame
      Get ComAdd of hoFrames "AK1,AK2" to voFrame
      Handle hoFrame
      Get Create (RefClass(cComFrame)) to hoFrame
      Set pvComObject of hoFrame to voFrame
         Set ComPadding of hoFrame OLEexPaddingAll to 8
         Set ComPadding of hoFrame OLEexPaddingBottom to 22
         Set ComBackColor of hoFrame to |CI$1000000
         Variant voPattern
         Get ComPattern of hoFrame to voPattern
         Handle hoPattern
         Get Create (RefClass(cComPattern)) to hoPattern
         Set pvComObject of hoPattern to voPattern
            Set ComType of hoPattern to OLEexPatternEmpty
         Send Destroy to hoPattern
         Set ComBackgroundExt of hoFrame to
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
      Send Destroy to hoFrame
   Send Destroy to hoFrames
   Send ComEndUpdate
End_Procedure
```

XBase++

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

```
PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oChartView
   LOCAL oFrame
   LOCAL oNodes

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

   oChartView := XbpActiveXControl():new( oForm:drawingArea )
   oChartView:CLSID  := "Exontrol.ChartView.1" /*{F4DFE455-01FE-420E-A088-
64346DCC3791}*/
   oChartView:create(,, {10,60},{610,370} )

      oChartView:BeginUpdate()
      oChartView:IndentSiblingY := 32

oChartView:VisualAppearance():Add(1,"gBFLBCJwBAEHhEJAADhABWMMACAADACAxF
 +;

"VIKGMIJTCiDpDliNQ+A+eB+CGCAiCiFgkgmT54HCdoGE0MhgkAM4LhOWJ/CQThlk4N
 +;

"BDiT4TnyKQ6nEbBjBeRhPnKGwYCkMYCgcGAnnGDBSD+RIHhKKJDDmMRSkSbg/nKY5
 +;
      "AwAggAIAYFQQ4WgEggDIAQgI=")
      oNodes := oChartView:Nodes()
         oNodes:Add("Child <b>1</b>",,"1234")
         oNodes:Add("Sub 1","1234","AK1")
         oNodes:Add("Sub 2","1234","AK2")
         oNodes:Add("Sub 3","1234")
         oNodes:Add("Child <b>2.1</b>"):AddGroup("Child <b>2.2</b>")
      oFrame := oChartView:Frames():Add("AK1,AK2")
         oFrame:SetProperty("Padding",-1/*exPaddingAll*/,8)
```

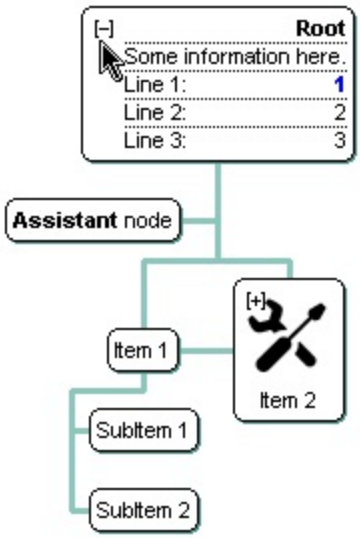
```
        oFrame:SetProperty("Padding",3/*exPaddingBottom*/,22)
        oFrame:SetProperty("BackColor",0x1000000)
        oFrame:Pattern():Type := 0/*exPatternEmpty*/
        oFrame:BackgroundExt :=
"top[4],left[4],right[4],client,bottom[4],bottom[16,back=0x01000000,text=`<sha
;;0>attention`,align=0x11]"
      oChartView:EndUpdate()

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

property ChartView.HasButtons as ExpandButtonEnum

Specifies whether a parent node displays +/- buttons if it contains child nodes.

Type	Description
ExpandButtonEnum	An ExpandButtonEnum expression that indicates whether the control display +/- buttons for nodes that contain child nodes.

By default, the HasButtons property is exNoButtons. The control displays no +/- buttons if the HasButtons property is exNoButtons. Use the HasButton property to hide a +/- button for a particular node. The node displays the +/- button only if the node contains child nodes, and the HasButton property is True. Use the Expanded property to expand or collapse a node by code. The control fires the Expand event when user expands or collapse a node. Use the ButtonsAlign property to specify the position of +/- buttons inside nodes. Use the HasButtonsCustom property to assign an icon for +/- buttons, when the HasButtons property is exCustom. Use the Images, ReplaceIcon methods to load new icons to the control's icons collection. Use the BeginUpdate and EndUpdate properties to maintain performance while do multiple changes to the control.



The following VB sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
With ChartView1
   .BeginUpdate
   .Images
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExm

   .HasButtons = exCustom
   .HasButtonsCustom(False) = 1
```

```
    .HasButtonsCustom(True) = 2
    .EndUpdate
End With
```

The following C++ sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
m_chartview.BeginUpdate();
m_chartview.Images(COleVariant("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZF

m_chartview.SetHasButtons( 4 /*exCustom*/ );
m_chartview.SetHasButtonsCustom( FALSE, 1 );
m_chartview.SetHasButtonsCustom( TRUE, 2 );
m_chartview.EndUpdate();
```

The following VB.NET sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
With AxChartView1

.Images("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlkt

    .HasButtons = EXORGCHARTLib.ExpandButtonEnum.exCustom
    .set_HasButtonsCustom(False, 1)
    .set_HasButtonsCustom(True, 2)
End With
```

The following C# sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
axChartView1.BeginUpdate();
axChartView1.Images("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA

axChartView1.HasButtons = EXORGCHARTLib.ExpandButtonEnum.exCustom;
axChartView1.set_HasButtonsCustom(false, 1);
axChartView1.set_HasButtonsCustom(true, 2);
axChartView1.EndUpdate();
```

The following VFP sample assigns different +/- buttons when expanding or collapsing the

parent nodes:

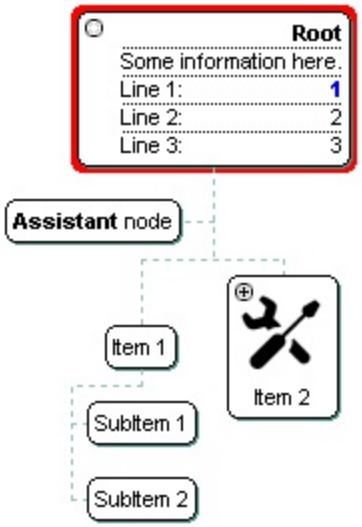
```
local s
s =
"gBJJgBAICAAEg4ACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjMLjABAAgjUYkUnlUrlktl0vmE

s = s +
"fkMZkkkjSAncPiUOjUTyOOjOSjLnZ+YjTvzjvjTuzuf0MZdAv0uizmoz0ZzurjGYzWVykYy202YA



with thisform.ChartView1
   .BeginUpdate()
   .Images(s)
   .HasButtons = 4 && exCustom
   local t
   t = "HasButtonsCustom(false) = 1" + chr(13) + chr(10)
   t = t + "HasButtonsCustom(true) = 2"
   .Template = t
   .EndUpdate
endwith
```

property ChartView.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 expanding state being changed. True value for- button, or False value for + button.
Long	A long expression that indicates the index of icon being displayed.

Use the HasButtonsCustom property to assign new appearance for +/- buttons, when HasButtons property is exCustom. Use the Images, ReplaceIcon methods to load new icons to the control's icons collection. Use the Expanded property to expand or collapse a node by code. The control fires the Expand event when user expands or collapse a node. Use the ButtonsAlign property to specify the position of +/- buttons inside nodes. Use the Image property to assign an icon to a node. Use the Picture property to load a picture to a node.



The following VB sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
With ChartView1
   .BeginUpdate
   .Images
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExm

   .HasButtons = exCustom
   .HasButtonsCustom(False) = 1
   .HasButtonsCustom(True) = 2
```

```
    .EndUpdate
End With
```

The following C++ sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
m_chartview.BeginUpdate();
m_chartview.Images(COleVariant("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZF

m_chartview.SetHasButtons( 4 /*exCustom*/ );
m_chartview.SetHasButtonsCustom( FALSE, 1 );
m_chartview.SetHasButtonsCustom( TRUE, 2 );
m_chartview.EndUpdate();
```

The following VB.NET sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
With AxChartView1

.Images("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlkt

    .HasButtons = EXORGCHARTLib.ExpandButtonEnum.exCustom
    .set_HasButtonsCustom(False, 1)
    .set_HasButtonsCustom(True, 2)
End With
```

The following C# sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
axChartView1.BeginUpdate();
axChartView1.Images("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA

axChartView1.HasButtons = EXORGCHARTLib.ExpandButtonEnum.exCustom;
axChartView1.set_HasButtonsCustom(false, 1);
axChartView1.set_HasButtonsCustom(true, 2);
axChartView1.EndUpdate();
```

The following VFP sample assigns different +/- buttons when expanding or collapsing the parent nodes:

```
local s
s =
"gBJJgBAICAAEg4ACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjMLjABAAgjUYkUnlUrlktl0vmE

s = s +
"fkMZkkkjSAncPiUOjUTyOOjOSjLnZ+YjTvzjvjTuzuf0MZdAv0uizmoz0ZzurjGYzWVykYy202YA



with thisform.ChartView1
   .BeginUpdate()
   .Images(s)
   .HasButtons = 4 && exCustom
   local t
   t = "HasButtonsCustom(false) = 1" + chr(13) + chr(10)
   t = t + "HasButtonsCustom(true) = 2"
   .Template = t
   .EndUpdate
endwith
```

property ChartView.hEBNList as Variant

Retrieves the handle of the skins list.

Type	Description
Variant	A long expression that specifies the handle of the skins list.

Reserved for internal use only.

property ChartView.hIconList as Variant

Retrieves the handle of the icons list.

Type	Description
Variant	A long expression that defines the handle of the icons list.

Reserved for internal use only.

property ChartView.HideSelection as Boolean

Specifies whether the selection in the control is hidden when the control loses the focus.

Type	Description
Boolean	A boolean expression that indicates whether the selection in the control is hidden when the control loses the focus.

By default, the HideSelection property is True. Use the SelectNode property to specify the selected node. The control fires the Select event when a node is selected. The SelColor property retrieves or sets a value that indicates the color used to mark the selected node. Use the DrawRoundNode property to draw round corners for the nodes. The ShadowNode property determines whether the control displays a shadow for nodes. Use the ShadowNode property to hide the shadow for a specific node.

property ChartView.hPictureList as Variant

Retrieves the handle of the pictures list.

Type	Description
Variant	A long expression that specifies the handle of the pictures list.

Reserved for internal use only.

property ChartView.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

Type	Description
Key as String	A String expression that indicates the key of the picture being added or replaced. If the Key property is Empty string, the entire collection of pictures is cleared.
Variant	The HTMLPicture specifies the picture being associated to a key. It can be one of the followings: • a string expression that indicates the path to the picture file, being loaded. • a string expression that indicates the base64 encoded string that holds a picture object, Use the eximages tool to save your picture as base64 encoded format. • A Picture object that indicates the picture being added or replaced. (A Picture object implements IPicture interface), If empty, the picture being associated to a key is removed. If the key already exists the new picture is replaced. If the key is not empty, and it doesn't not exist a new picture is added.

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> tags. By default, the HTMLPicture collection is empty. Use the HTMLPicture property to add new pictures to be used in HTML captions. For instance, the HTMLPicture("pic1") = "c:\winnt\zapotec.bmp", loads the zapotec picture and associates the pic1 key to it. Any "<img>pic1</img>" sequence in HTML captions, displays the pic1 picture. On return, the HTMLPicture property retrieves a Picture object (this implements the IPictureDisp interface). The HTMLPicture property supports: BMP, EMF, EXIF, GIF, ICON, JPEG, PNG, TIFF or WMF formats. A picture being shown in the HTML captions can be displayed using a different size, by specifying the size in the second parameter of the <img> tag as <img>pic1:18</pic> means that the pic1 is displayed using a different size. Use the Picture property to assign a custom- sized picture to a node.

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"
```

```
<NODE1>.Caption = "A <img>pic1</img>"
<NODE2>.Caption = "B <img>pic2</img>"
<NODE3>.Caption = "A <img>pic1</img> + B <img>pic2</img>"
```

property ChartView.hWnd as Long

Retrieves the control's window handle.

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

The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

method ChartView.Images (Handle as Variant)

Sets at runtime the control's image list. The Handle should be a handle to an Image List Control.

Type	Description
Handle as Variant	The Handle parameter can be: • A string expression that specifies the ICO file to add. The ICO file format is an image file format for computer icons in Microsoft Windows. ICO files contain one or more small images at multiple sizes and color depths, such that they may be scaled appropriately. For instance, Images("c:\temp\copy.ico") method adds the sync.ico file to the control's Images collection *(string, loads the icon using its path)* • A string expression that indicates the BASE64 encoded string that holds the icons list. Use the Exontrol's ExImages tool to save/load your icons as BASE64 encoded format. In this case the string may begin with "gBJJ..." *(string, loads icons using base64 encoded string)* • A reference to a Microsoft ImageList control (mscomctl.ocx, MSComctlLib.ImageList type) that holds the icons to add *(object, loads icons from a Microsoft ImageList control)* • A reference to a Picture (IPictureDisp implementation) that holds the icon to add. For instance, the VB's LoadPicture (Function LoadPicture([FileName], [Size], [ColorDepth], [X], [Y]) As IPictureDisp) or LoadResPicture (Function LoadResPicture(id, restype As Integer) As IPictureDisp) returns a picture object (object, loads icon from a Picture object) • A long expression that identifies a handle to an Image List Control (the Handle should be of HIMAGELIST type). On 64-bit platforms, the Handle parameter must be a Variant of LongLong / LONG_PTR data type (signed 64-bit (8-byte) integers), saved under llVal field, as VT_I8 type. The LONGLONG / LONG_PTR is __int64, a 64-bit integer. For instance, in C++ you can use as Images(COleVariant((LONG_PTR)hImageList)) or Images(COleVariant(

(LONGLONG)hImageList)), where hImageList is of HIMAGELIST type. The GetSafeHandle() method of the CImageList gets the HIMAGELIST handle (long, loads icon from HIMAGELIST type)

The user can add images at design time, by drag and drop files to combo's image holder. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. Use the Image property to assign an icon to a node. Use the Picture property to load a picture to a node.

The following VB sample uses the Microsoft Image List control:

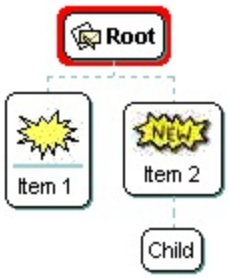
```
ChartView1.Images ImageList1.hImageList
```

The following VB sample loads icons and pictures using the BASE64 encoded strings:

```
Dim s As String
With ChartView1
   .BackColor = vbWhite

   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8pIUrlktI0vmEx

   With .Root
      .Image = 1
   End With

   With .Nodes.Add("Item 1")
      s =
"gBCJr+BAAg0HGwEgwog4jg4ig4BAEFg4AZEKisZjUbAAzg5mg6Zg7Mg7/g0ek8oGcgjsijsk

      s = s +
"XgBadIDXdYSXRb9wWBcIK2taF1gAI5HiPaN8oPdINWbaF23KAwyWkNYyXxg9p3WNYjU/c

      .Picture = s
   End With
```

```
    With .Nodes.Add("Item 2", , "ketA")
        s =
"gBCJoqBAAg0HGwEgwog4qg4Xg4BAEFg4AegDisZjUbgwzg5mg6Zg7Mg7/jsHGceAAzkEr

        s = s +
"XU5WtD2VbVlFZUwDAAGQD3La4yjIAADAOQQrAOFYc2IGRA1nTwAVgAASXGKwVi7fgADI

        s = s +
"K7lDLYABKAaArVsisUuoWK6m1UKyYIpSVihRrPSIILKQi51OK6USqJUDIlSzBU4LNSKjVDytlll

        .Picture = s
    End With

    With .Nodes.Add("Child", "ketA")
    End With

End With
```

Run the sample and you get:



The following C++ sample loads a collection of icons from a BASE64 encoded string:

```
m_chartview.Images(COleVariant("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjA
 );
m_chartview.GetRoot().SetImage( 1 );
```

The following C++ sample loads the system's image list (used by Windows Explorer):

```
SHFILEINFO sfi; ZeroMemory( &sfi, sizeof(sfi) );
m_chartView.Images( COleVariant( (long)SHGetFileInfo(_T("C:\\"),  0, &sfi, sizeof
```

```
(SHFILEINFO), SHGFI_SMALLICON | SHGFI_SYSICONINDEX  ) ) );
```

If you are using the CImageList class, you should use the HIMAGELIST operator of the CImageList class, to pass to the Images method like described:

```
m_chartView.Images( COleVariant( long( pImageList->operator HIMAGELIST() ) ) );
```

The following VB.NET sample loads a collection of icons from a BASE64 encoded string:

```
With AxChartView1

.Images("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0pIUrlk

   .Root.Image = 1
End With
```

The following C# sample loads a collection of icons from a BASE64 encoded string:

```
axChartView1.Images("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAE

axChartView1.Root.Image = 1;
```

The following VFP sample loads a collection of icons from a BASE64 encoded string:

```
With thisform.ChartView1
   local s
   s =
"gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0pIUrlktl0vmExr

   s = s +
"8jN+o17tFSumBx2PpWDutSxVhqGIqGVs+QzmdneYu0XxVr0GLvuZvOe1Wrmul1mv2GP0u

   .Images(s)
   .Root.Image = 1
EndWith
```

property ChartView.ImageSize as Long

Retrieves or sets the size of icons the control displays..

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 node displays
- expand/collapse glyphs

property ChartView.IndentChild as Long

Retrieves or sets the amount, in pixels, that child nodes are indented relative to their parent nodes.

Type	Description
Long	A long expression that indicates the amount, in pixels, that child nodes are indented relative to their parent nodes.

By default, the IndentChild property is 15 pixels. The IndentChild property has effect only for nodes whose ArrangeSiblingNodesAs property is exTree. The PenLink property defines the width of the pen used to paint the links between nodes. If the PenLink property is larger than 1, its value counts when the control indents the child nodes relative to their parent node. Use the IndentSiblingY property to indent vertically the sibling nodes. Use the IndentSiblingX property to indent horizontally the sibling nodes.

property ChartView.IndentSiblingX as Long

Specifies the horizontal distance, in pixels between two siblings node.

Type	Description
Long	A long expression that specifies the horizontal distance, in pixels between two siblings node.

By default, the IndentSiblingX property is 15 pixels. The PenLink property defines the width of the pen used to paint the links between nodes. If the PenLink property is larger than 1, its value counts when the control indents horizontally the sibling nodes. Use the IndentSiblingY property to indent vertically the sibling nodes. The IndentChild property retrieves or sets the amount, in pixels, that child nodes are indented relative to their parent nodes, which has effect if the ArrangeSiglingNodesAs property is exTree. The Layout property indicates whether the nodes are arranged from Top to Bottom (TTB) or Left to Right (LTR).

The indent properties work differently based on the Layout property as follows:

- **exLayoutTTB** (by default) - the nodes are arranged from top to the bottom:
  - IndentSiblingX property indicates the horizontal distance between two sibling nodes.
  - IndentSiblingY property indicates the vertical distance between a node and it's child nodes.
  - InflateGroupY property indicates the vertical distance to increase the group of nodes (AddGroup).

- **exLayoutLTR** - the nodes are arranged from left to the right:
  - IndentSiblingX property indicates the horizontal distance between a node and it's child nodes.
  - IndentSiblingY property indicates the vertical distance between two sibling nodes.
  - InflateGroupX property indicates the horizontal distance to increase the group of nodes (AddGroup).

property ChartView.IndentSiblingY as Long

Specifies the vertical distance, in pixels between two siblings node.

Type	Description
Long	A long expression that specifies the vertical distance, in pixels between two siblings node.

By default, the IndentSiblingY property is 15 pixels. The PenLink property defines the width of the pen used to paint the links between nodes. If the PenLink property is larger than 1, its value counts when the control indents vertically the sibling nodes. Use the IndentSiblingX property to indent horizontally the sibling nodes. The IndentChild property retrieves or sets the amount, in pixels, that child nodes are indented relative to their parent nodes, which has effect if the ArrangeSiglingNodesAs property is exTree.

The indent properties work differently based on the Layout property as follows:

- **exLayoutTTB** (by default) - the nodes are arranged from top to the bottom:
  - IndentSiblingX property indicates the horizontal distance between two sibling nodes.
  - IndentSiblingY property indicates the vertical distance between a node and it's child nodes.
  - InflateGroupY property indicates the vertical distance to increase the group of nodes (AddGroup).

- **exLayoutLTR** - the nodes are arranged from left to the right:
  - IndentSiblingX property indicates the horizontal distance between a node and it's child nodes.
  - IndentSiblingY property indicates the vertical distance between two sibling nodes.
  - InflateGroupX property indicates the horizontal distance to increase the group of nodes (AddGroup).

property ChartView.Layout as ChartLayoutEnum

Specifies the way the chart displays the diagram.

Type	Description
ChartLayoutEnum	A ChartLayoutEnum expression that indicates whether the nodes are arranged from Top to Bottom (TTB) or Left to Right (LTR).

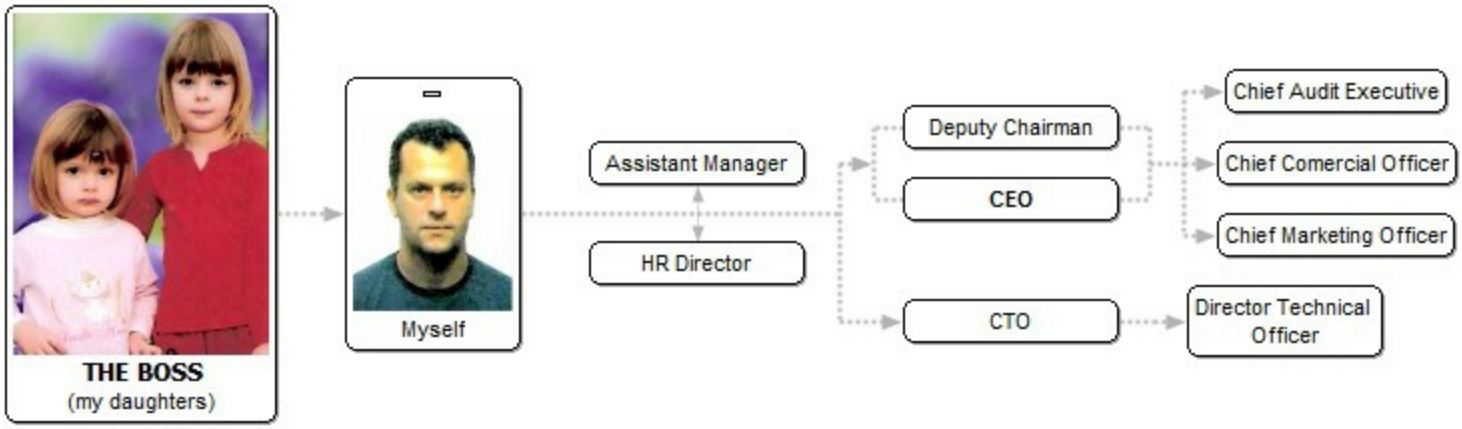
By default, the Layout property is exLayoutTTB. Use the Layout property to arrange the nodes from left to right. If the Layout property is exLayoutTTB (by default), the parent nodes go from the top to bottom, the child nodes go from left to right. If Layout property is exLayoutLTR, the parent nodes go from the left to right, the child nodes go from top to bottom.

The indent properties work differently based on the Layout property as follows:

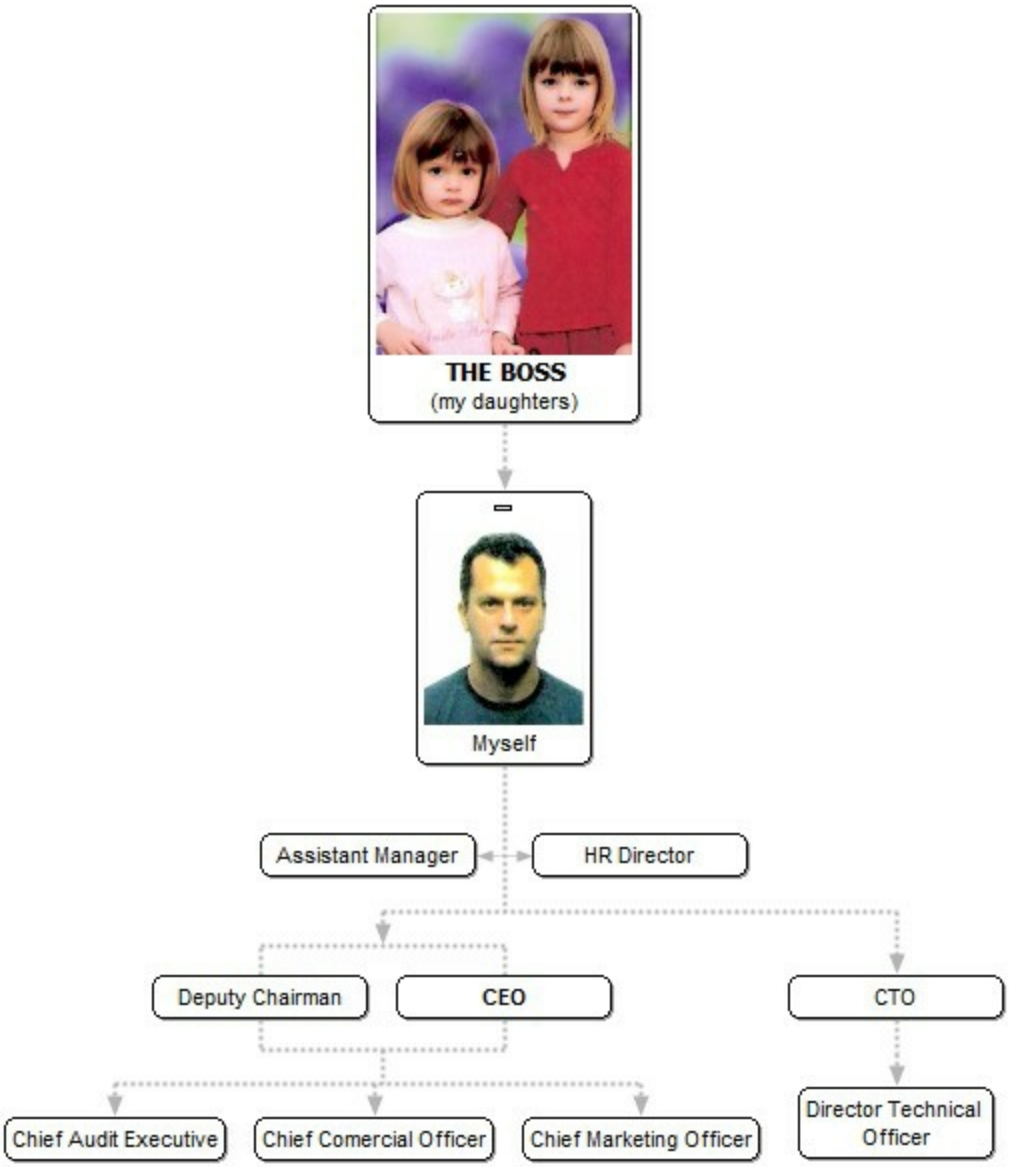
- **exLayoutTTB** (by default) - the nodes are arranged from top to the bottom:
  - IndentSiblingX property indicates the horizontal distance between two sibling nodes.
  - IndentSiblingY property indicates the vertical distance between a node and it's child nodes.
  - InflateGroupY property indicates the vertical distance to increase the group of nodes (AddGroup).

- **exLayoutLTR** - the nodes are arranged from left to the right:
  - IndentSiblingX property indicates the horizontal distance between a node and it's child nodes.
  - IndentSiblingY property indicates the vertical distance between two sibling nodes.
  - InflateGroupX property indicates the horizontal distance to increase the group of nodes (AddGroup).

The following screen shot shows the nodes arrangement when the Layout property is exLayoutLTR:



The following screen shot shows the nodes arrangement when the Layout property is exLayoutTTB (by default):



property ChartView.LinkAssistantColor as Color

Specifies the color for assistant links.

Type	Description
Color	A color expression that indicates the color for links between assistant nodes.

If the LinkAssistantColor property is equal with BackColor property no links are painted. Use the PenLink property to define the pen used to paint the links between nodes. Use the PenLinkAssistant property to define the type of the pen used to paint the links between assistant nodes. For instance, if the LinkAssistantColor property has the same value as BackColor property, the control doesn't paint the links between the nodes. Use the PenWidthLinkAssistant property to specify the thickness of the links between assistant nodes. Use the LinkColor property to specify the color for links between nodes.

property ChartView.LinkCaptionFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Node

Gets the node whose caption on the link hovers the specified 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
Node	A Node object whose LinkCaption property hovers the specified point.

Use the LinkCaptionFromPoint property to determine the node whose caption on the link is at specified point. **If the X parameter is -1 and Y parameter is -1 the LinkCaptionFromPoint property determines the caption from the cursor**. Use the LinkCaption property to assign a HTML caption on the node's link.

The following VB sample prints the caption on the link from the cursor:

```
Private Sub ChartView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .LinkCaptionFromPoint(-1,-1)
      If Not (n Is Nothing) Then
         Debug.Print n.LinkCaption
      End If
   End With
End Sub
```

The following C++ sample prints the caption on the link from the cursor:

```
void OnMouseMoveChartview1(short Button, short Shift, long X, long Y)
{
   CNode node = m_chartview.GetLinkCaptionFromPoint( -1, -1 );
   if ( node.m_lpDispatch != NULL )
      OutputDebugString( node.GetLinkCaption() );
```

```
}
```

The following VB.NET sample prints the caption on the link from the cursor:

```
Private Sub AxChartView1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent) Handles
AxChartView1.MouseMoveEvent
   With AxChartView1
      Dim n As EXORGCHARTLib.Node = .get_LinkCaptionFromPoint(-1, -1)
      If Not (n Is Nothing) Then
         Debug.WriteLine(n.LinkCaption)
      End If
   End With
End Sub
```

The following C# sample prints the caption on the link from the cursor:

```
private void axChartView1_MouseMoveEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent e)
{
   EXORGCHARTLib.Node node = axChartView1.get_LinkCaptionFromPoint(-1, -1);
   if (node != null)
      System.Diagnostics.Debug.WriteLine(node.LinkCaption);
}
```

The following VFP sample prints the caption on the link from the cursor:

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

With thisform.ChartView1
   local n
   n = .LinkCaptionFromPoint(-1 , -1 )
   If !isnull(n) Then
      wait window nowait n.LinkCaption
   EndIf
EndWith
```

property ChartView.LinkColor as Color

Specifies the color for links.

Type	Description
Color	A color expression that indicates the color for links between nodes.

If the LinkColor property is equal with BackColor property no links are painted. Use the LinkAssistantColor property to specify the color of the links between assistant nodes. Use the PenLink property to define the pen used to paint the links between nodes. For instance, the LinkColor property has the same value as BackColor property, the control doesn't paint the links between the nodes. Use the PenWidthLink property to specify the thickness of the links between nodes.

property ChartView.LinkToColor as Color

Specifies the color for 'LinkTo' links.

Type	Description
Color	A Color expression that specifies the color to show the links between nodes, being added using the LinkTo property.

The LinkToColor property specifies the color of the links between nodes, being added using the LinkTo property. The LinkToColor property specifies a different color to be used for an arbitrary link. The following properties specify the style, width and the color for links between nodes, being added using the LinkTo property:

- PenLinkTo property specifies the style of the link to be shown between nodes.
- PenWidthLinkTo property specifies the size / width of the links between nodes.
- LinkToColor property specifies the color to show the links between nodes.

The AntiAliasing property specifies whether smoothing (antialiasing) is applied to lines or curves in the control. The ShowLinksDir property specifies whether the links shows the direction between the nodes.

method ChartView.LoadXML (Source as Variant)

Loads an XML document from the specified location, using MSXML parser.

Type	Description
Source as Variant	An indicator of the object that specifies the source for the XML document. The object can represent a file name, a URL, an IStream, a SAFEARRAY, or an IXMLDOMDocument.

Return	Description
Boolean	A boolean expression that specifies whether the XML document is loaded without errors. If an error occurs, the method retrieves a description of the error occurred.

The LoadXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to fetch the control's data from XML documents, previously saved using the SaveXML method. The control is emptied when the LoadXML method is called, and so the nodes collection is emptied before loading a new XML document. The **<root>** element holds information about the root node of the control, including its child nodes stored in **<node>** elements. Properties like Caption, Key, Image, ImageAlignment, Expanded or Left for assistant nodes, are saved for each node. The **<assistants>** element contains a collection of **<assistant>** elements that holds information about an assistant node.

The XML format looks like follows:

```
- <Content Author Component Version>
  - <Root Key Caption Image ImageAlignment Expanded>
    - <Node Key Caption Image ImageAlignment Expanded>
      - <Node Key Caption Image ImageAlignment Expanded/>
      - <Node Key Caption Image ImageAlignment Expanded>
        - <Node Key Caption Image ImageAlignment Expanded/>
        - <Node Key Caption Image ImageAlignment Expanded>
          </Node>
        ....
        - <Assistants>
            <Assistant Caption Left Image ImageAlignment />
          </Assistants>
       </Node>
      ....
      - <Assistants>
```

```
            <Assistant Caption Left Image ImageAlignment />
          </Assistants>
        </Node>
      - <Assistants>
          <Assistant Caption Left Image ImageAlignment />
        </Assistants>
    </Root>
</Content>
```

property ChartView.MaxZoomHeight as Double

Gets or sets a value indicating how large the chart will appear on vertical axis (max value).

Type	Description
Double	A Double expression that specifies the upper limit on vertical axis to resize the chart.

The LayoutStartChanging(exResizeChart) event notifies your application once the user starts resizing the chart. The LayoutEndChanging(exResizeChart) event notifies your application once the chart is resized. The ZoomHeight property specifies a value that indicates how large the chart will appear on vertical axis. Use the MinZoomWidth/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. Use the MinZoomHeight/MaxZoomHeight property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. The AllowResizeChart property specifies the combination of keys so the user can resize the chart at runtime,

property ChartView.MaxZoomWidth as Double

Gets or sets a value indicating how large the chart will appear on horizontal axis (max value).

Type	Description
Double	A Double expression that specifies the upper limit on horizontal axis to resize the chart.

The LayoutStartChanging(exResizeChart) event notifies your application once the user starts resizing the chart. The LayoutEndChanging(exResizeChart) event notifies your application once the chart is resized. The ZoomHeight property specifies a value that indicates how large the chart will appear on vertical axis. Use the MinZoomWidth/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. Use the MinZoomHeight/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. The AllowResizeChart property specifies the combination of keys so the user can resize the chart at runtime,

property ChartView.MinZoomHeight as Double

Gets or sets a value indicating how large the chart will appear on vertical axis (min value).

Type	Description
Double	A Double expression that specifies the lower limit on vertical axis to resize the chart.

The LayoutStartChanging(exResizeChart) event notifies your application once the user starts resizing the chart. The LayoutEndChanging(exResizeChart) event notifies your application once the chart is resized. The ZoomHeight property specifies a value that indicates how large the chart will appear on vertical axis. Use the MinZoomWidth/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. Use the MinZoomHeight/MaxZoomHeight property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. The AllowResizeChart property specifies the combination of keys so the user can resize the chart at runtime,

property ChartView.MinZoomWidth as Double

Gets or sets a value indicating how large the chart will appear on horizontal axis (min value).

Type	Description
Double	A Double expression that specifies the lower limit on horizontal axis to resize the chart.

The LayoutStartChanging(exResizeChart) event notifies your application once the user starts resizing the chart. The LayoutEndChanging(exResizeChart) event notifies your application once the chart is resized. The ZoomHeight property specifies a value that indicates how large the chart will appear on vertical axis. Use the MinZoomWidth/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. Use the MinZoomHeight/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking. The AllowResizeChart property specifies the combination of keys so the user can resize the chart at runtime,

property ChartView.NodeFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Node

Gets the node 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
Node	A Node object where the point is.

Use the NodeFromPoint property to determine the node from specified position. **If the X parameter is -1 and Y parameter is -1 the NodeFromPoint property determines the node from the cursor**. Use the Caption property to specify the caption of the node. The control fires the Select event when the user clicks a node.

The following VB sample prints the caption of the node from the cursor:

```
Private Sub ChartView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .NodeFromPoint(-1,-1)
      If Not (n Is Nothing) Then
         Debug.Print n.Caption
      End If
   End With
End Sub
```

The following C++ sample prints the caption of the node from the cursor:

```
void OnMouseMoveChartview1(short Button, short Shift, long X, long Y)
{
   CNode node = m_chartview.GetNodeFromPoint( -1, -1 );
   if ( node.m_lpDispatch != NULL )
      OutputDebugString( node.GetCaption() );
}
```

The following VB.NET sample prints the caption of the node from the cursor:

```
Private Sub AxChartView1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent) Handles
AxChartView1.MouseMoveEvent
   With AxChartView1
      Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(-1, -1)
      If Not (n Is Nothing) Then
         Debug.WriteLine(n.Caption)
      End If
   End With
End Sub
```

The following C# sample prints the caption of the node from the cursor:

```
private void axChartView1_MouseMoveEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent e)
{
   EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(-1, -1);
   if (node != null)
      System.Diagnostics.Debug.WriteLine(node.Caption);
}
```

The following VFP sample prints the caption of the node from the cursor:

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

With thisform.ChartView1
   local n
   n = .NodeFromPoint(-1 , -1 )
   If !isnull(n) Then
      wait window nowait n.Caption
   EndIf
EndWith
```

property ChartView.Nodes as Nodes

Gets the control's collection of nodes.

Type	Description
Nodes	A Nodes object that holds the control's nodes collection.

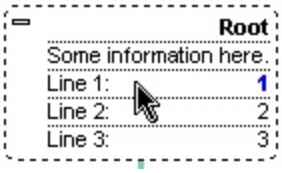
Use the Nodes property to access the control's nodes collection. Use the Add method to add new nodes to organigram. Use the Remove method to remove a specific node. Use the Item property to retrieve a node in the Nodes collection. Use the BeginUpdate and EndUpdate methods to maintain performance while adding new nodes. Use the Images method to assign a collection of icons to the control. Use the Root property to retrieve the root node. The Frames property of the control gives access to the Frames collection.

property ChartView.PenBorderNode as PenTypeEnum

Specifies the type of pen used to draw the node's borders.

Type	Description
PenTypeEnum	A PenTypeEnum expression that indicates the type of pen used to paint the borders for nodes.

Use the PenBorderNode property to define the type of the pen being used to paint the borders of the nodes. Use the PenBorderNode property to define the type of the pen used to paint the borders for a specified node. Use the ShadowNode property to hide the node's shadow. Use the DrawRoundNode property to specify define round corners for all nodes in the organigram.

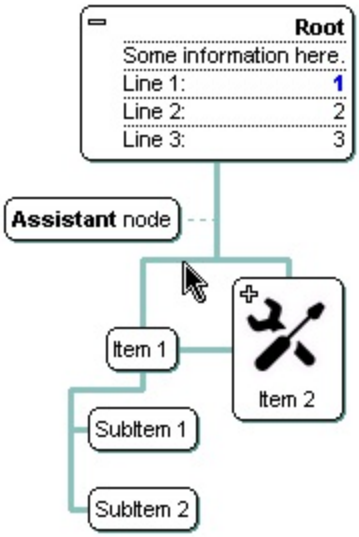


property ChartView.PenLink as PenTypeEnum

Specifies the type of the pen used to paint the links between nodes.

Type	Description
PenTypeEnum	A PenTypeEnum expression that indicates the type of the pen used to paint the links between nodes.

By default, the PenLink property is exPenDot. Use the PenLink property to define the type of the pen used to paint the links between nodes. Use the PenLinkAssistant property to define the type of the pen used to paint the links between assistant nodes. Use the LinkColor property to hide the links between nodes. Use the PenWidthLink property to specify the thickness of the links between nodes. For instance, the LinkColor property has the same value as BackColor property, the control doesn't paint the links between the nodes. Use the PenBorderNode property to specify the pent used to paint the borders of the node.

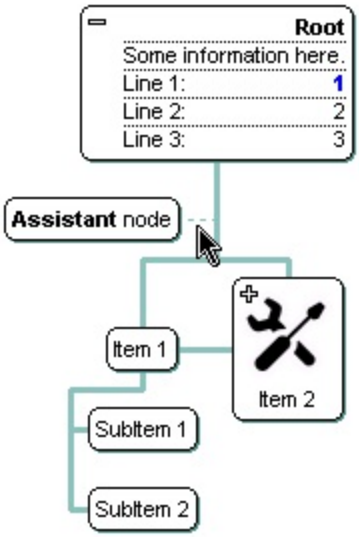


property ChartView.PenLinkAssistant as PenTypeEnum

Specifies the type of the pen used to paint the links between assistant nodes.

Type	Description
PenTypeEnum	A PenTypeEnum expression that indicates the type of the pen used to paint the links between assistant nodes.

By default, the PenLinkAssistant property is exPenDot. Use the PenLinkAssistant property to define the type of the pen used to paint the links between assistant nodes. Use the AddAssistant property to add an assistant node. The LinkAssistantColor property specifies the color of the links between assistant nodes. Use the PenWidthLinkAssistant property to specify the thickness of the links between assistant nodes. For instance, the LinkAssistantColor property has the same value as BackColor property, the control doesn't paint the links between assistant nodes. Use the PenLink property to define the type of the pen used to paint the links between nodes.



property ChartView.PenLinkTo as PenTypeEnum

Specifies the type of the pen used to show the 'LinkTo' links.

Type	Description
PenTypeEnum	A PenTypeEnum expression that specifies the link to be shown between nodes, being added using the LinkTo property

By default, the PenLinkTo property is exPenDot. Use the PenLinkTo property to specify the style of the links between nodes, being added using the LinkTo property. The LinkToPen property specifies the pen to show an linkto line. The following properties specify the style, width and the color for links between nodes, being added using the LinkTo property:

- PenLinkTo property specifies the style of the link to be shown between nodes.
- PenWidthLinkTo property specifies the size / width of the links between nodes.
- LinkToColor property specifies the color to show the links between nodes.

The AntiAliasing property specifies whether smoothing (antialiasing) is applied to lines or curves in the control. The ShowLinksDir property specifies whether the links shows the direction between the nodes.

property ChartView.PenWidthLink as Long

Specifies the width of the links between nodes.

Type	Description
Long	A long expression that specifies the width (in pixels) of the links between nodes.

By default, the PenWidthLink property is 1. Use the PenWidthLink property to specify the thickness of the lines between nodes. Use the PenLink property to specify the type of the pen used to paint the lines between nodes. By default, the PenWidthLinks property is 1. The LinkColor property specifies the color of the links between nodes.

property ChartView.PenWidthLinkAssistant as Long

Specifies the width of the links between assistant nodes.

Type	Description
Long	A long expression that specifies the width (in pixels) of the links between assistant nodes.

By default, the PenWidthLinkAssistant property is 1. Use the PenWidthLinkAssistant property to specify the thickness of the lines between nodes. Use the PenLinkAssistant property to specify the type of the pen used to paint the lines between assistant nodes. The LinkAssistantColor property specifies the color of the links between assistant nodes. Use the PenWidthLink property to specify the width of the link between nodes.

property ChartView.PenWidthLinkTo as Long

Specifies the width of the 'LinkTo' links.

Type	Description
Long	A long expression that specifies the size of the link between nodes, being added using the LinkTo property.

By default, the PenWidthLinkTo property is 1. Use the PenWidthLinkTo property to specify the width of the links between nodes, being added using the LinkTo property. The LinkToWidth property specifies the width for an linkto line. The following properties specify the style, width and the color for links between nodes, being added using the LinkTo property:

- PenLinkTo property specifies the style of the link to be shown between nodes.
- PenWidthLinkTo property specifies the size / width of the links between nodes.
- LinkToColor property specifies the color to show the links between nodes.

The AntiAliasing property specifies whether smoothing (antialiasing) is applied to lines or curves in the control. The ShowLinksDir property specifies whether the links shows the direction between the nodes.

property ChartView.Picture as IPictureDisp

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

Type	Description
IPictureDisp	A Picture object being displayed on the control's background.

Use the Picture property to display a picture on the control's background. Use the PictureDisplay property to align the picture on the control's background. Use the BackColor property to define the control's background color. Use the BackColorNode property to define the default background color for nodes. Use the BackColor property to define the background color for a specified node. Use the SelColor property to specify the color to mark the selected node. Use the SelectNode property to specify the selected node.

property ChartView.PictureAspectRatioNode as AspectRatioEnum

Specifies the default aspect ratio for the node's picture.

Type	Description
AspectRatioEnum	An ApsectRatioEnum expression that specifies the aspect ratio for each picture in the nodes collection.

By default, the PictureAspectRatioNode property is exAspectRatioNone. If the PictureAspectRatioNode property is set, it is applied to all nodes, excepts where the node's PictureAspectRatio is being set. In other words, you can use the PictureAspectRatioNode property to apply the same aspect ratio for all pictures in the chart. If using the PictureAspectRatio property you can use the PictureWidthNode or PictureHeightNode property to specify the fixed width or height for the picture. Use the Picture property to specify the node's picture.

The aspect ratio for the node's picture works as follows:

- **exAspectRatioWidth**, the PictureWidth or PictureWidthNode property must specify the fixed width, so the height of the displaying picture is calculated based on the original size, so it keeps its aspect ratio.
- **exAspectRatioHeight**, the PictureHeight or PictureHeightNode property must specify the fixed height, so the width of the displaying picture is calculated based on the original size, so it keeps its aspect ratio.

The size of the picture being shown in the node is computed as follow:

- if the PictureWidth and/or PictureHeight property is specified, it indicates the width or the height of the picture being shown in the node.
- if the PictureWidthNode and/or PictureHeightNode properties of the ChartView object is specified, it indicates the width or the height of the picture being shown in the node.
- If none of these is specified, the node displays the picture using its size.

In any case, the size of the picture is also influenced by the the aspect ratio for the node's picture as it is specified by PictureAspectRatio or PictureAspectRatioNode property while any of these is not exAspectRatioNone.

property ChartView.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 specifes the alignment of the picture on the control's background.

The PictureDisplay property aligns the Picture on the control's background. The PictureDisplay property has no effect if the Picture property is empty. Use the BackColor property to define the control's background color. Use the SelColor property to specify the color to mark the selected node. Use the SelectNode property to specify the selected node.

property ChartView.PictureHeightNode as Long

Specifies the height of the node's picture.

Type	Description
Long	A long expression that specifies the height to display the pictures in nodes. If not specified, the picture's height is used instead.

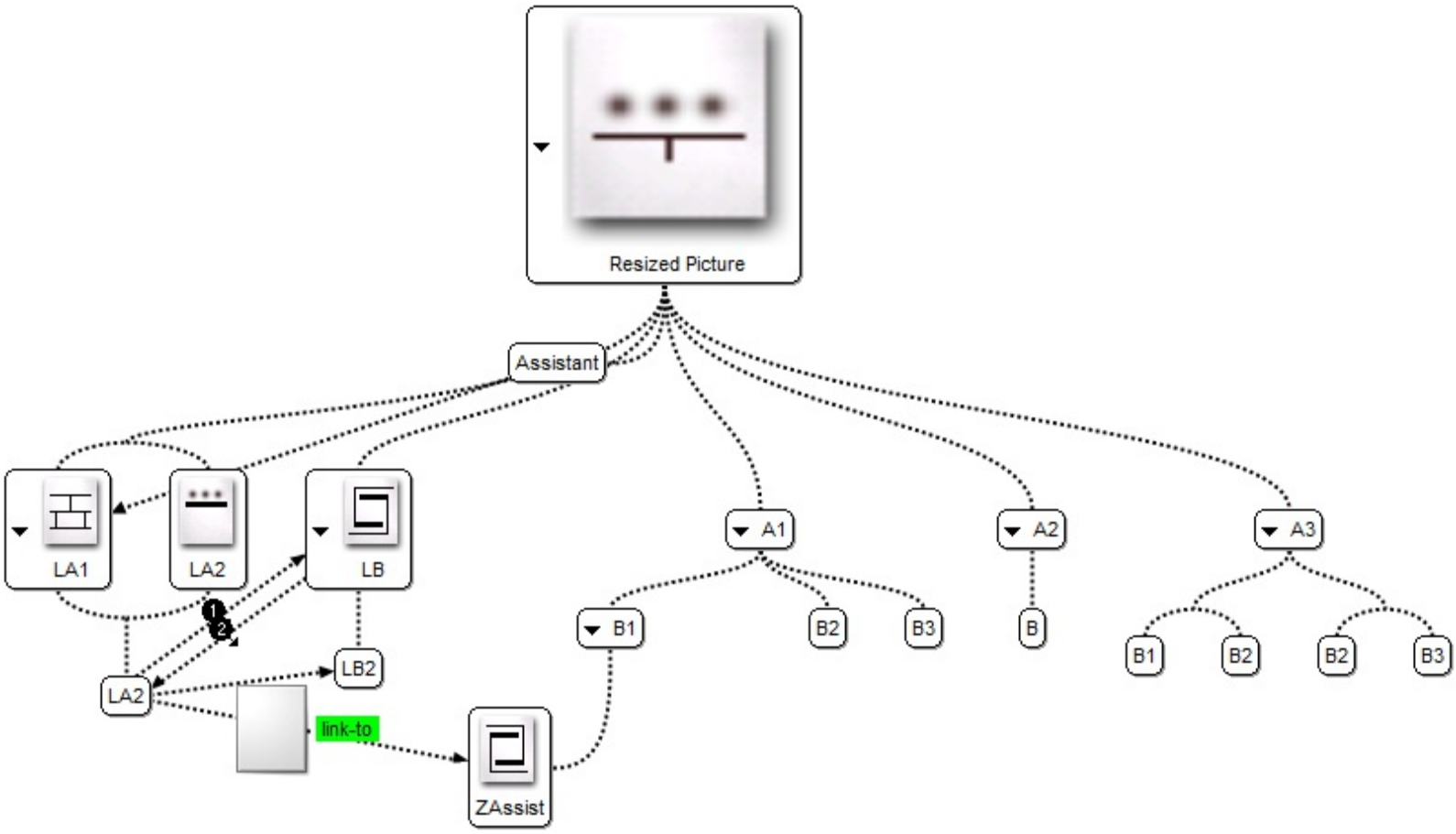
By default, the PictureHeightNode property is -1. Use the PictureHeightNode property to specify a different height to display all pictures in all nodes. Use the Picture property to specify the node's picture. The PictureWidthNode and PictureHeightNode properties of the ChartView object handles the size of the displayed pictures for all the nodes. The PictureAspectRatioNode property specifies the aspect ratio for the node's picture based on the original width or height. The PictureWidth and PictureHeight properties handles the size of displayed picture for specified node.

The size of the picture being shown in the node is computed as follow:

- if the PictureWidth and/or PictureHeight property is specified, it indicates the width or the height of the picture being shown in the node.
- if the PictureWidthNode and/or PictureHeightNode properties of the ChartView object is specified, it indicates the width or the height of the picture being shown in the node.
- If none of these is specified, the node displays the picture using its size.

In any case, the size of the picture is also influenced by the the aspect ratio for the node's picture as it is specified by PictureAspectRatio or PictureAspectRatioNode property while any of these is not exAspectRatioNone.

The following screen shot shows the root's picture resized, while the other nodes display the pictures using their original size.


Resized Picture
Assistant
LA1
LA2
LB
A1
A2
A3
B1
B2
B3
B
B1
B2
B2
B3
LA2
LB2
link-to
ZAssist

property ChartView.PictureWidthNode as Long

Specifies the width of the node's picture.

Type	Description
Long	A long expression that specifies the width to display the pictures in nodes. If not specified, the picture's width is used instead.

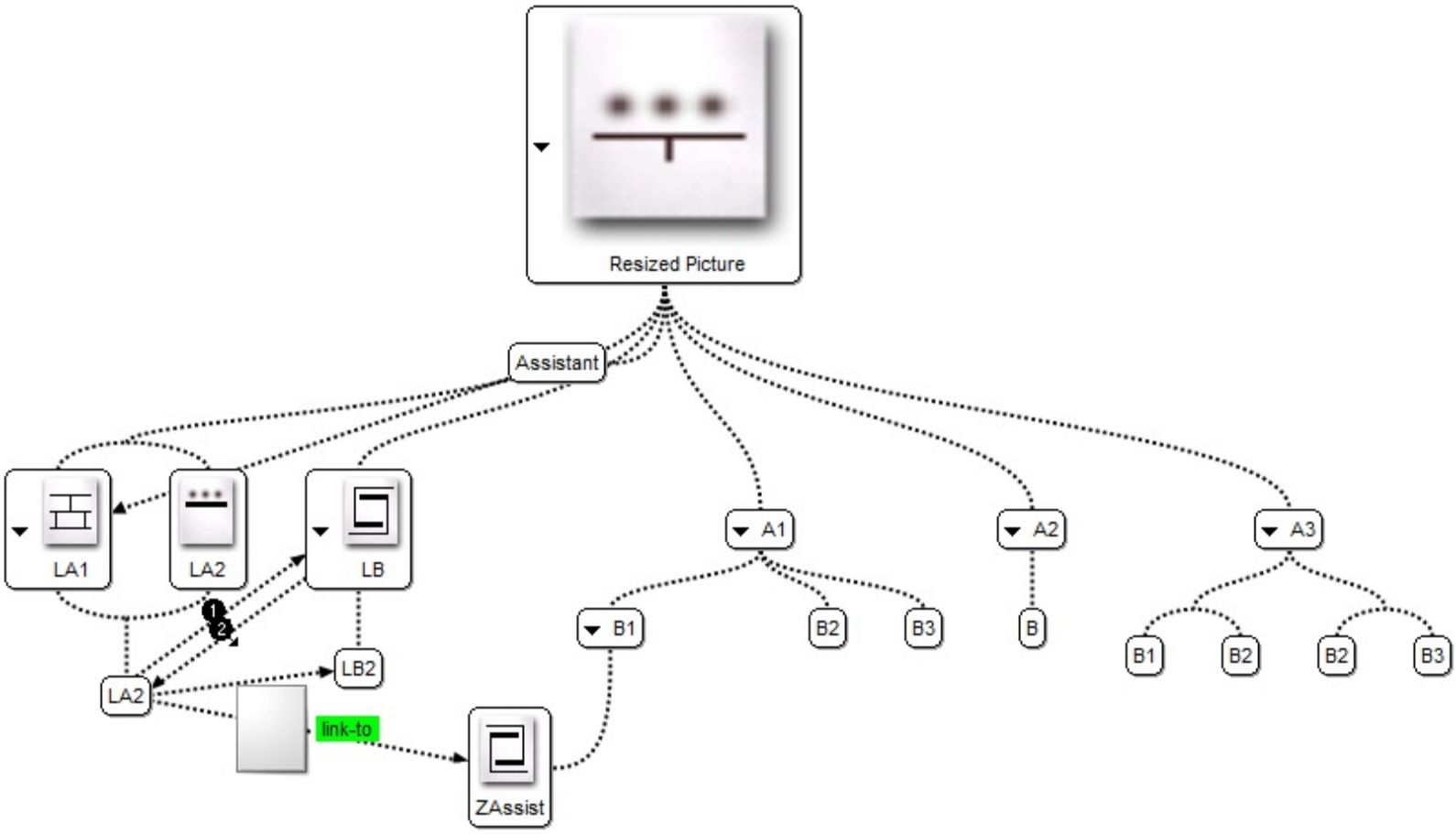
By default, the PictureWidthNode property is -1. Use the PictureWidthNode property to specify a different width to display all pictures in all nodes. Use the Picture property to specify the node's picture. The PictureWidthNode and PictureHeightNode properties of the ChartView object handles the size of the displayed pictures for all the nodes. The PictureAspectRatioNode property specifies the aspect ratio for the node's picture based on the original width or height. The PictureWidth and PictureHeight properties handles the size of displayed picture for specified node.

The size of the picture being shown in the node is computed as follow:

- if the PictureWidth and/or PictureHeight property is specified, it indicates the width or the height of the picture being shown in the node.
- if the PictureWidthNode and/or PictureHeightNode properties of the ChartView object is specified, it indicates the width or the height of the picture being shown in the node.
- If none of these is specified, the node displays the picture using its size.

In any case, the size of the picture is also influenced by the the aspect ratio for the node's picture as it is specified by PictureAspectRatio or PictureAspectRatioNode property while any of these is not exAspectRatioNone.

The following screen shot shows the root's picture resized, while the other nodes display the pictures using their original size.


Resized Picture
Assistant
LA1
LA2
LB
A1
A2
A3
B1
B2
B3
B
B1
B2
B2
B3
LA2
LB2
link-to
ZAssist

method ChartView.Refresh ()

Refreshes the control.

Type	Description

The Refresh method refreshes the control. The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Use the Font property to specify the control's font. Use the hWnd property to get the handle of the control's window.

The following VB sample calls the Refresh method:

```
ChartView1.Refresh
```

The following C++ sample calls the Refresh method:

```
m_chartview.Refresh();
```

The following VB.NET sample calls the Refresh method:

```
AxChartView1.CtlRefresh()
```

In VB.NET the System.Windows.Forms.Control class has already a Refresh method, so the CtlRefresh method should be called.

The following C# sample calls the Refresh method:

```
axChartView1.CtlRefresh();
```

In C# the System.Windows.Forms.Control class has already a Refresh method, so the CtlRefresh method should be called.

The following VFP sample calls the Refresh method:

```
thisform.ChartView1.Object.Refresh()
```

method ChartView.ReplaceIcon ([Icon as Variant], [Index as Variant])

Adds a new icon, replaces an icon or clears the control's image list.

Type	Description
Icon as Variant	A long expression that indicates the icon's handle.
Index as Variant	A long expression that indicates the index where icon is inserted.
Return	**Description**
Long	A long expression that indicates the index of the icon in the images collection

Use the ReplaceIcon property to add, remove or replace an icon in the control's images collection. Also, the ReplaceIcon property can clear the images collection. Use the Images method to attach a image list to the control.

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

i = ExChartView1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added

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

i = ExChartView1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.

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

ExChartView1.ReplaceIcon 0, i, i specifies the index of icon removed.

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

ExChartView1.ReplaceIcon 0, -1

property ChartView.Root as Node

Gets the root node.

Type	Description
Node	A Node object that indicates the root node of the organigram.

Use the Root property to access the root node of the organigram. Use the Nodes property to access the nodes of the organigram. The root node cannot be removed. By default, the root's caption is "Root". Use the Caption property to change the node's caption. The Root's Key property is "root". Use the Item property to access an item by its key. Use the Image property to assign an icon to a node. Use the Picture property to assign a custom size picture to a node. Use the BeginUpdate

The following VB sample assigns a caption and a picture to the root node:

```
With ChartView1
    .BeginUpdate
      With .Root
        .Caption = "<r><dotline><b>Root</b><br>Second line of the root"
        .Picture = "c:\winnt\system32\n2k.bmp"
      End With
    .EndUpdate
End With
```

The following C++ sample assigns a caption and a picture to the root node:

```
m_chartview.BeginUpdate();
CNode node = m_chartview.GetRoot();
node.SetCaption( "<r><dotline><b>Root</b><br>Second line of the root" );
node.SetPicture( COleVariant( "c:\\winnt\\system32\\n2k.bmp" ) );
m_chartview.EndUpdate();
```

The following VB.NET sample assigns a caption and a picture to the root node:

```
With AxChartView1
  .BeginUpdate()
  With .Root
    .Caption = "<r><dotline><b>Root</b><br>Second line of the root"
```

```
        .Picture = "c:\winnt\system32\n2k.bmp"
    End With
    .EndUpdate()
End With
```

The following C# sample assigns a caption and a picture to the root node:

```
axChartView1.BeginUpdate();
EXORGCHARTLib.Node node = axChartView1.Root;
node.Caption = "<r><dotline><b>Root</b><br>Second line of the root";
node.Picture = "c:\\winnt\\system32\\n2k.bmp";
axChartView1.EndUpdate();
```

The following VFP sample assigns a caption and a picture to the root node:

```
With thisform.ChartView1
    .BeginUpdate
        With .Root
            .Caption = "<r><dotline><b>Root</b><br>Second line of the root"
            .Picture = "c:\winnt\system32\n2k.bmp"
        EndWith
    .EndUpdate
EndWith
```

method ChartView.SaveXML (Destination as Variant)

Saves the control's content as XML document to the specified location, using the MSXML parser.

Type	Description

Destination as Variant	This object can represent a file name, reference to a string member, an XML document object, or a custom object that supports persistence as follows:
	
	• String - Specifies the file name. Note that this must be a file name, rather than a URL. The file is created if necessary and the contents are entirely replaced with the contents of the saved document. For example:
	
	`ChartView1.SaveXML("sample.xml")`
	
	• Reference to a String member - Saves the control's content to the string member. Note that the string member must be empty, before calling the SaveXML method. For example:
	
	`Dim s As String`
	`ChartView1.SaveXML s`
	
	In VB.NET for /NET assembly, you should call such as :
	
	`Dim s As String = String.Empty`
	`Exchartview1.SaveXML(s)`
	
	In C# for /NET assembly, you should call such as :
	
	`string s = string.Empty;`
	`exchartview1.SaveXML(ref s);`
	
	• XML Document Object. For example:
	
	`Dim xmldoc as Object`
	`Set xmldoc = CreateObject("MSXML.DOMDocument")`
	`ChartView1.SaveXML(xmldoc)`
	
	• Custom object supporting persistence - Any other custom COM object that supports **QueryInterface** for **IStream**, **IPersistStream**, or **IPersistStreamInit** can also be provided here and the document will be saved accordingly. In the **IStream** case, the **IStream::Write**

method will be called as it saves the document; in the **IPersistStream** case, **IPersistStream::Load** will be called with an **IStream** that supports the **Read**, **Seek**, and **Stat** methods.

Return	Description
Boolean	A Boolen expression that specifies whether saving the XML document was ok.

The SaveXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to save the control's data to XML documents. Use the LoadXML method to load XML documents, previously saved with the SaveXML method. The control is emptied when the LoadXML method is called, and so the nodes collection is emptied before loading a new XML document. The **<root>** element holds information about the root node of the control, including its child nodes stored in **<node>** elements. Properties like Caption, Key, Image, ImageAlignment, Expanded or Left for assistant nodes, are saved for each node. The **<assistants>** element contains a collection of **<assistant>** elements that holds information about an assistant node.

The XML format looks like follows:

```
- <Content Author Component Version>
  - <Root Key Caption Image ImageAlignment Expanded>
    - <Node Key Caption Image ImageAlignment Expanded>
      - <Node Key Caption Image ImageAlignment Expanded/>
      - <Node Key Caption Image ImageAlignment Expanded>
        - <Node Key Caption Image ImageAlignment Expanded/>
        - <Node Key Caption Image ImageAlignment Expanded>
          </Node>
        ....
        - <Assistants>
            <Assistant Caption Left Image ImageAlignment />
          </Assistants>
        </Node>
      ....
      - <Assistants>
          <Assistant Caption Left Image ImageAlignment />
        </Assistants>
    </Node>
  - <Assistants>
```

```
        <Assistant Caption Left Image ImageAlignment />
      </Assistants>
    </Root>
</Content>
```

property ChartView.ScrollButtonHeight as Long

Specifies the height of the button in the vertical scrollbar.

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

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

property ChartView.ScrollButtonWidth as Long

Specifies the width of the button in the horizontal scrollbar.

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

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

property ChartView.ScrollByClick as Boolean

Specifies a value that indicates whether the user scrolls the control's content by clicking the client area.

Type	Description
Boolean	A boolean expression that indicates whether the user scrolls the control's content by clicking the client area.

By default, the ScrollByClick property is True. If the ScrollByClick property is True, the user can scroll the control's content (if the control's scrollbars are visible), if the user clicks the control's client area and drag the cursor to a new position. Use The ScrollPos property to programmatically scroll the chart. Use the EnsureVisibleNode property to ensure that a node fits the control's client area. Use the SelectNode property to select a node.

property ChartView.ScrollFont (ScrollBar as ScrollBarEnum) as IFontDisp

Retrieves or sets the scrollbar's font.

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

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

property ChartView.ScrollHeight as Long

Specifies the height of the horizontal scrollbar.

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

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

method ChartView.ScrollOnCursor (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Scrolls the chart as the cursor indicates.

Type	Description
X as OLE_XPOS_PIXELS	A long expression that specifies the screen x-coordinate of the cursor, -1 for the current cursor position
Y as OLE_YPOS_PIXELS	A long expression that specifies the screen y-coordinate of the cursor, -1 for the current cursor position

ScrollOnCursor method scrolls the chart based on the position of the cursor. So, if the cursor is on the top of the chart, the control get scrolled up, else if the cursor is on the bottom side of the chart, the chart gets scrolled down, and so on. The BorderWidth and BorderHeight properties specifies the width and height of the borders of the chart. For instance, you can use the ScrollOnCursor method to scroll during ole drag and drop operation, as you can see in the next sample. The ScrollOnCursor method scrolls the chart pixel by pixel, so you can call several times, so multiple pixels are scrolled once. Use the BeginUpdate/EndUpdate methods to maintain performance while calling the ScrollOnCursor several times. Use the ScrollPos property to programmatically scroll the chart.

The following C# sample scrolls the chart if the cursor is on the borders of the chart, while drag and drop:

```
private void chartview1_DragOver(object sender, DragEventArgs e)
{
   chartview1.BeginUpdate();
   for ( int i = 0; i < 4; i++ )
      chartview1.ScrollOnCursor();
   chartview1.EndUpdate();
   e.Effect = DragDropEffects.None;
   exontrol.EXORGCHARTLib.Node spNode = chartview1.get_NodeFromPoint(-1, -1);
   if (spNode != null)
      if ( spNode.Key != e.Data.GetData(DataFormats.Text).ToString() )
         e.Effect = e.AllowedEffect;
}
```

The following VB.NET sample scrolls the chart if the cursor is on the borders of the chart, while drag and drop:

```
Private Sub Chartview1_DragOver(ByVal sender As System.Object, ByVal e As
```

```
System.Windows.Forms.DragEventArgs) Handles Chartview1.DragOver
    With Chartview1
        .BeginUpdate()
        For i As Integer = 1 To 4
            .ScrollOnCursor()
        Next
        .EndUpdate()
    End With
    e.Effect = DragDropEffects.None
    Dim spNode As exontrol.EXORGCHARTLib.Node = Chartview1.get_NodeFromPoint(-1,
-1)
    If Not spNode Is Nothing Then
        If spNode.Key <> e.Data.GetData(DataFormats.Text).ToString() Then
            e.Effect = e.AllowedEffect
        End If
    End If
End Sub
```

property ChartView.ScrollOnEnsure as Boolean

Specifies a value that indicates whether the control scrolls the control's content when ensuring that a node is visible.

Type	Description
Boolean	A boolean expression that indicates whether the control scrolls the control's content when ensuring that a node is visible.

By default, the ScrollOnEnsure property is True. Use the EnsureVisibleNode property to ensure that a node fits the control's client area. Use the SelectNode property to select a node. The control automatically scrolls the control's content to ensure that the node being clicked fits the control's client area, if the EnsureVisibleOnSelect property is True. Use the ScrollByClick property to specify whether the control's content is scrolled to a new position when the user clicks and drags the cursor inside the control's client area.

property ChartView.ScrollOrderParts(ScrollBar as ScrollBarEnum) as String

Specifies the order of the buttons in the scroll bar.

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

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

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

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

The list of supported literals in the ScrollOrderParts property is:

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

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

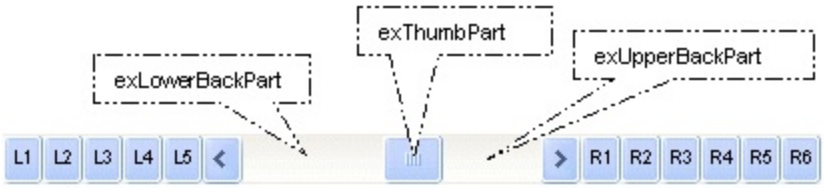
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

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

Specifies the caption being displayed on the specified scroll part.

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

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



By default, the following parts are shown:

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

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

```
With ChartView1
  .BeginUpdate
```

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

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

```
With AxChartView1
    .BeginUpdate()
    .set_ScrollPartVisible(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part Or
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
    .EndUpdate()
End With
```

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

```
axChartView1.BeginUpdate();
axChartView1.set_ScrollPartVisible(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part |
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, true);
axChartView1.set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axChartView1.set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axChartView1.EndUpdate();
```

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

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

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

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

```
With thisform.ChartView1
   .BeginUpdate
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

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

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

property ChartView.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.

The following VB sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
With ChartView1
   .ScrollPartCaption(exHScroll,exLowerBackPart) = "left"
   .ScrollPartCaptionAlignment(exHScroll,exLowerBackPart) = LeftAlignment
   .ScrollPartCaption(exHScroll,exUpperBackPart) = "right"
   .ScrollPartCaptionAlignment(exHScroll,exUpperBackPart) = RightAlignment
   .FixedWidthNode = 320
End With
```

The following VB.NET sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
With AxChartView1

.set_ScrollPartCaption(EXORGCHARTLib.ScrollBarEnum.exHScroll,EXORGCHARTLib.ScrollPar



.set_ScrollPartCaptionAlignment(EXORGCHARTLib.ScrollBarEnum.exHScroll,EXORGCHARTLi
```

```
.set_ScrollPartCaption(EXORGCHARTLib.ScrollBarEnum.exHScroll,EXORGCHARTLib.ScrollPar

.set_ScrollPartCaptionAlignment(EXORGCHARTLib.ScrollBarEnum.exHScroll,EXORGCHARTLi

    .FixedWidthNode = 320
End With
```

The following C# sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
axChartView1.set_ScrollPartCaption(EXORGCHARTLib.ScrollBarEnum.exHScroll,EXORGCHAR

axChartView1.set_ScrollPartCaptionAlignment(EXORGCHARTLib.ScrollBarEnum.exHScroll,EX

axChartView1.set_ScrollPartCaption(EXORGCHARTLib.ScrollBarEnum.exHScroll,EXORGCHAR

axChartView1.set_ScrollPartCaptionAlignment(EXORGCHARTLib.ScrollBarEnum.exHScroll,EX

axChartView1.FixedWidthNode = 320;
```

The following C++ sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

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

    #import "C:\\WinNT\\System32\\ExOrgChart.dll"
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1-
```

```
>PutScrollPartCaption(EXORGCHARTLib::exHScroll,EXORGCHARTLib::exLowerBackPart,L"lef

spChartView1-
>PutScrollPartCaptionAlignment(EXORGCHARTLib::exHScroll,EXORGCHARTLib::exLowerBac

spChartView1-
>PutScrollPartCaption(EXORGCHARTLib::exHScroll,EXORGCHARTLib::exUpperBackPart,L"rig

spChartView1-
>PutScrollPartCaptionAlignment(EXORGCHARTLib::exHScroll,EXORGCHARTLib::exUpperBac

spChartView1->PutFixedWidthNode(320);
```

The following VFP sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

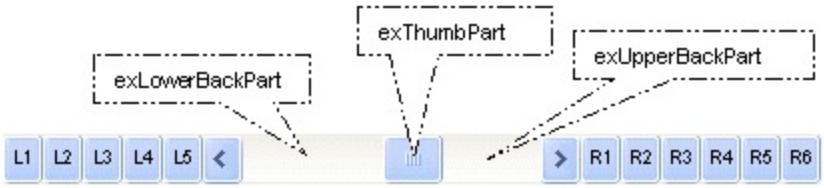
```
with thisform.ChartView1
   .ScrollPartCaption(1,512) = "left"
   .ScrollPartCaptionAlignment(1,512) = 0
   .ScrollPartCaption(1,128) = "right"
   .ScrollPartCaptionAlignment(1,128) = 2
   .FixedWidthNode = 320
endwith
```

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

Indicates whether the specified scroll part is enabled or disabled.

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

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

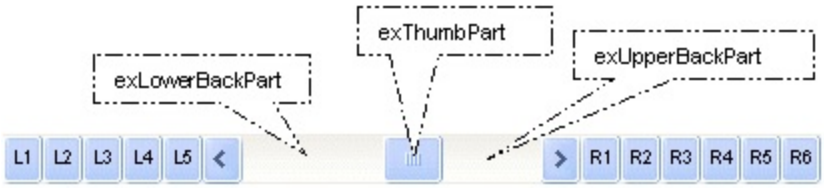


property ChartView.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is visible or hidden.

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

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



By default, the following parts are shown:

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

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

```
With ChartView1
  .BeginUpdate
```

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

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

```
With AxChartView1
    .BeginUpdate()
    .set_ScrollPartVisible(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part Or
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
    .EndUpdate()
End With
```

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

```
axChartView1.BeginUpdate();
axChartView1.set_ScrollPartVisible(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part |
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, true);
axChartView1.set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axChartView1.set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axChartView1.EndUpdate();
```

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

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

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

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

```
With thisform.ChartView1
   .BeginUpdate
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

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

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

property ChartView.ScrollPos(Vertical as Boolean) as Long

Specifies the vertical/horizontal scroll position.

Type	Description
Vertical as Boolean	A boolean expression that specifies the scrollbar being requested. True indicates the Vertical scroll bar, False indicates the Horizontal scroll bar.
Long	A long expression that specifies the scroll bar position.

Use the ScrollPos property to programmatically scroll the chart. Use the ScrollPos property to set or get the horizontal or vertical scroll position. Use the ScrollOrderParts property to specify the order of the buttons in the chart's scrollbar. Use the ScrollPartCaption property to assign a caption to a button in the scroll bar. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. The ScrollOnCursor method scrolls the control's chart based on the position of the cursor.

property ChartView.ScrollThumbSize(ScrollBar as ScrollBarEnum) as Long

Specifies the size of the thumb in the scrollbar.

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

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

property ChartView.ScrollToolTip(ScrollBar as ScrollBarEnum) as String

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

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

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

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

```
Private Sub ChartView1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As
Long)
   If (Not Horizontal) Then
      ChartView1.ScrollToolTip(exVScroll) = "Record " & NewVal
   End If
End Sub
```

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

```
Private Sub AxChartView1_OffsetChanged(ByVal sender As System.Object, ByVal e As
AxEXCHARTVIEWLib._IChartViewEvents_OffsetChangedEvent) Handles
AxChartView1.OffsetChanged
   If (Not e.horizontal) Then
      AxChartView1.set_ScrollToolTip(EXCHARTVIEWLib.ScrollBarEnum.exVScroll, "Record "
& e.newVal.ToString())
   End If
End Sub
```

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

```
void OnOffsetChangedChartView1(BOOL Horizontal, long NewVal)
{
   if ( !Horizontal )
   {
      CString strFormat;
      strFormat.Format( _T("%i"), NewVal );
      m_chartView.SetScrollToolTip( 0, strFormat );
   }
}
```

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

```
private void axChartView1_OffsetChanged(object sender,
AxEXCHARTVIEWLib._IChartViewEvents_OffsetChangedEvent e)
{
   if ( !e.horizontal )
      axChartView1.set_ScrollToolTip(EXCHARTVIEWLib.ScrollBarEnum.exVScroll, "Record "
+ e.newVal.ToString());
}
```

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

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

If (1 # horizontal) Then
   thisform.ChartView1.ScrollToolTip(0) = "Record " + ltrim(str(newval))
EndIf
```

property ChartView.ScrollWidth as Long

Specifies the width of the vertical scrollbar.

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

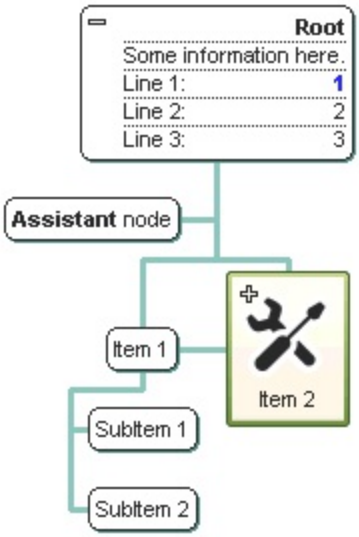
By default, the ScrollWidth property is -1. If the ScrollWidth property is -1, the control uses the default width of the vertical scroll bar from the system. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons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

property ChartView.SelColor as Color

Retrieves or sets a value that indicates the color used to mark the selected node.

Type	Description
Color	A color expression that indicates the color used to mark the selected node. 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.

A disabled node is painted as non selected. Use the SelectNode property to determine the selected node. For instance, if the SelColor property has the same value as BackColor property, the control doesn't paint the mark around the selected node. The control fires the Select event when a node is selected.



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

```
With ChartView1
   .VisualAppearance.Add 1, "D:\Temp\ExOrgChart.Help\select.ebn"
   .SelColor = &H1000000
End With
```

The following C++ sample changes the visual appearance for the selected node:

```
#include "Appearance.h"
m_chartview.GetVisualAppearance().Add( 1,
COleVariant("D:\\Temp\\ExOrgChart.Help\\select.ebn") );
m_chartview.SetSelColor( 0x1000000 );
```

The following VB.NET sample changes the visual appearance for the selected node:

```
With AxChartView1
   .VisualAppearance.Add(1, "D:\Temp\ExOrgChart.Help\select.ebn")
   .Template = "SelColor = 16777216"
End With
```

The following C# sample changes the visual appearance for the selected node:

```
axChartView1.VisualAppearance.Add(1, "D:\\Temp\\ExOrgChart.Help\\select.ebn");
axChartView1.Template = "SelColor = 16777216";
```

The following VFP sample changes the visual appearance for the selected node:

```
With thisform.ChartView1
   .VisualAppearance.Add(1, "D:\Temp\ExOrgChart.Help\select.ebn")
   .SelColor = 16777216
EndWith
```

The following VB sample changes the background and foreground color for the selected node:

```
Private Sub ChartView1_Select(ByVal OldNode As EXORGCHARTLibCtl.INode, ByVal
NewNode As EXORGCHARTLibCtl.INode)
   If Not (OldNode Is Nothing) Then
      With OldNode
         .ClearBackColor
         .ClearForeColor
      End With
   End If
   With NewNode
      .ForeColor = vbWhite
      .BackColor = vbBlue
```

```
    End With
End Sub
```

The following C++ sample changes the background and foreground color for the selected node:

```
void OnSelectChartview1(LPDISPATCH OldNode, LPDISPATCH NewNode)
{
    CNode oldNode( OldNode ); oldNode.m_bAutoRelease = FALSE;
    CNode newNode( NewNode ); newNode.m_bAutoRelease = FALSE;

    if ( oldNode.m_lpDispatch != NULL )
    {
        oldNode.ClearBackColor();
        oldNode.ClearForeColor();
    }
    newNode.SetBackColor( RGB(0,0,128) );
    newNode.SetForeColor( RGB(255,255,255) );
}
```

The following VB.NET sample changes the background and foreground color for the selected node:

```
Private Sub AxChartView1_SelectEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_SelectEvent) Handles AxChartView1.SelectEvent
    If Not (e.oldNode Is Nothing) Then
        With e.oldNode
            .ClearBackColor()
            .ClearForeColor()
        End With
    End If
    With e.newNode
        .ForeColor = ToUInt32(Color.White)
        .BackColor = ToUInt32(Color.Blue)
    End With
End Sub
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background and foreground color for the selected node:

```
private void axChartView1_SelectEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_SelectEvent e)
{
   if ( e.oldNode != null )
   {
      e.oldNode.ClearBackColor();
      e.oldNode.ClearForeColor();
   }
   e.newNode.BackColor = ToUInt32(Color.Blue);
   e.newNode.ForeColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background and foreground color for the selected node:

```
*** ActiveX Control Event ***
LPARAMETERS oldnode, newnode
```

```
If !isnull(oldnode)
   With oldnode
      .ClearBackColor
      .ClearForeColor
   EndWith
EndIf
With newnode
   .ForeColor = RGB(255,255,255)
   .BackColor = RGB(0,0,128)
EndWith
```

property ChartView.SelectNode as Variant

Specifies the selected node.

Type	Description
Variant	A Node object that indicates the selected node or a string expression that indicates the key of the node being selected.

Use the SelectNode property to retrieve or sets the selected node. The control fires the Select event when a node is selected. The SelColor property retrieves or sets a value that indicates the color used to mark the selected node. Use the DrawRoundNode property to draw round corners for the nodes. The ShadowNode property determines whether the control displays a shadow for nodes. Use the ShadowNode property to hide the shadow for a specific node. Use the Key property to determine the key of the node. Use the Caption property to specify the caption of the node.

The following VB sample selects the node with the key "key":

```
ChartView1.SelectNode = "key"
```

The following VB sample prints the caption of the selected node:

```
With ChartView1
   Debug.Print .SelectNode.Caption
End With
```

The following C++ sample selects the node with the key "key":

```
m_chartview.SetSelectNode( COleVariant( "key" ) );
```

The following C++ sample prints the caption of the selected node:

```
CNode node( V_DISPATCH( &m_chartview.GetSelectNode() ) );
OutputDebugString( node.GetCaption() );
```

The following VB.NET sample selects the node with the key "key":

```
With AxChartView1
   .SelectNode = "key"
End With
```

The following VB.NET sample prints the caption of the selected node:

```
With AxChartView1
   Debug.WriteLine(.SelectNode.Caption())
End With
```

The following C# sample selects the node with the key "key":

```
axChartView1.SelectNode = "key";
```

The following C# sample prints the caption of the selected node:

```
EXORGCHARTLib.Node node = axChartView1.SelectNode as EXORGCHARTLib.Node;
System.Diagnostics.Debug.WriteLine(node.Caption);
```

The following VFP sample selects the node with the key "key":

```
With thisform.ChartView1
   .SelectNode = "key"
EndWith
```

The following VFP sample prints the caption of the selected node:

```
With thisform.ChartView1
   wait window nowait .SelectNode.Caption
EndWith
```

property ChartView.ShadowNode as Boolean

Specifies whether the node has shadow.

Type	Description
Boolean	A boolean expression that indicates whether the nodes displays its shadow.

The ShadowNode property determines whether the control displays a shadow for nodes. Use the ShadowNode property to hide the shadow for a specific node. Use the DrawRoundNode property to specify whether the node has borders with round corners. Use the DrawRoundNode property to specify define round corners for all nodes in the organigram. The DrawRoundNode property and ShadowNode property has effect only if no skin is applied to a node. Use the Background property to specify a background color or a visual appearance for specific parts in the control.

property ChartView.ShowAddNew as Boolean

Specifies whether the selected node shows or hides add new buttons.

Type	Description
Boolean	A boolean expression that specifies whether the selected node shows or hides add new buttons.

By default, the ShowAddNew property is False. The ShowAddNew property shows or hides the add new button for selected nodes.

property ChartView.ShowAssistants as Boolean

Retrieves or sets a value that indicates whether the assistant nodes are shown.

Type	Description
Boolean	A boolean expression that indicates whether the assistant nodes are visible or hidden.

Use the ShowAssistants property to hide the assistant nodes. Use the AddAssistant method to add assistant nodes. By default, the ShowAssistants property is True. The control displays the assistant nodes only if the ShowAssistants property is True. Use the Assistant property to access the assistant nodes collection. Use the RemoveAssistant method to remove an assistant node.

property ChartView.ShowImageList as Boolean

Specifies whether the control's image list window is visible or hidden.

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

By default, the ShowImageList property is True. Use the ShowImageList property to hide the control's images list window. The control's images list window is visible only at design time. Use the Images method to associate an images list control to the tree control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. Use the Image property to assign an icon to a node. Use the Picture property to load a picture to a node.

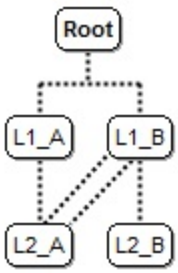
property ChartView.ShowLinksDir as Boolean

Specifies whether links show the direction.

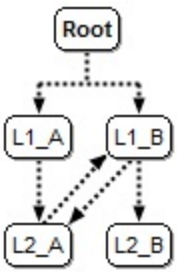
Type	Description
Boolean	A Boolean expression that specifies whether the links between nodes show the direction.

By default, the ShowLinksDir property is False. Use the ShowLinksDir property to show the direction of the links. The LinkTo property adds arbitrary a link between any two nodes. The LinkToCaption property specifies the HTML caption being shown on the links between nodes. The LinkColor property specifies the color of the links between nodes. The ShowLinks property doesn't affect the links added with the LinkTo property. The AntiAliasing property specifies whether the control uses the antialiasing rendering to show the arrows of the links. The ShowRoundLink property specifies whether the round links are shown between parent and child nodes. Currently, the direction of the link is shown only for rectangular links, not for the round links. The PenWidthLink property specifies the thickness of the lines between nodes. Use the PenLink property to specify the type of the pen used to paint the lines between nodes. In case, the arrows of the link are not shown use the IndentSiblingY, IndentSiblingX or IndentChild property to increase the distance between sibling or child nodes. The LinkToShowDir property specifies the whether the links shows its direction . If the ShowLinksDir property is True, you can use the ShowLinkDir property to specify whether the node should show or hide its direction.

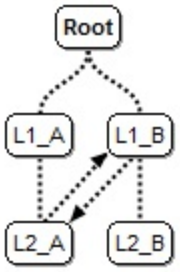
The following screen shot shows the links between nodes with no direction:



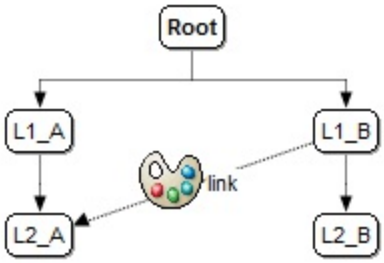
The following screen shot shows the links between nodes with their direction:



The following screen shot shows the links between nodes with their direction, and ShowRoundLink property on True:



The following screen shot shows the links between nodes with their direction including HTML captions on links:



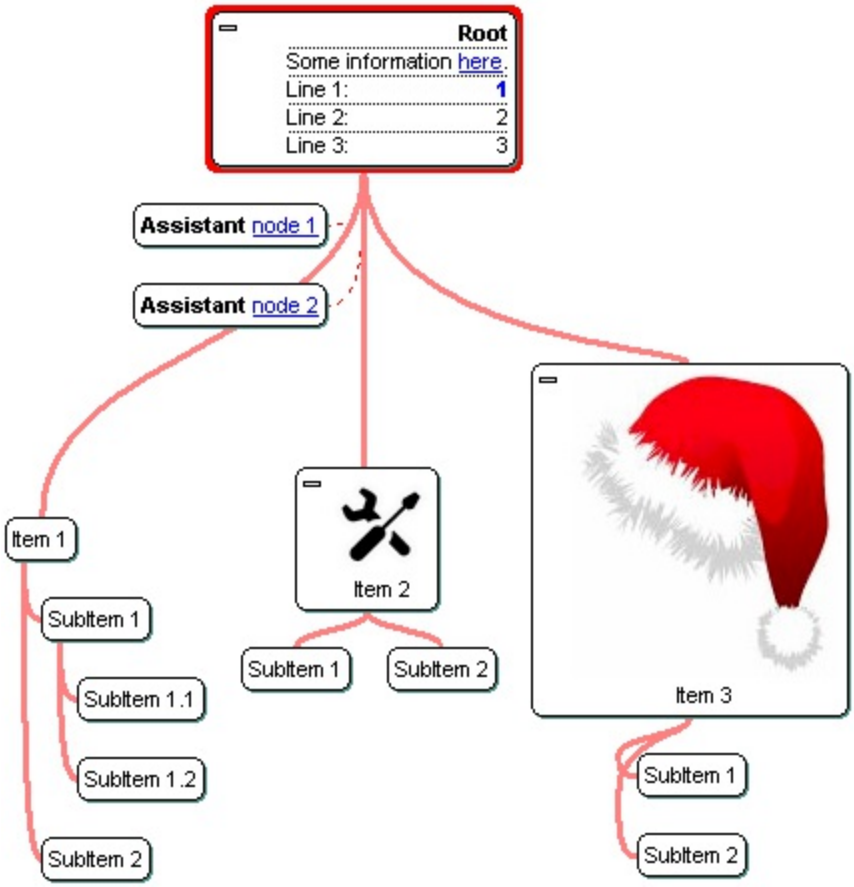
property ChartView.ShowRoundLink as Boolean

Specifies whether the round links are shown between parent and child nodes.

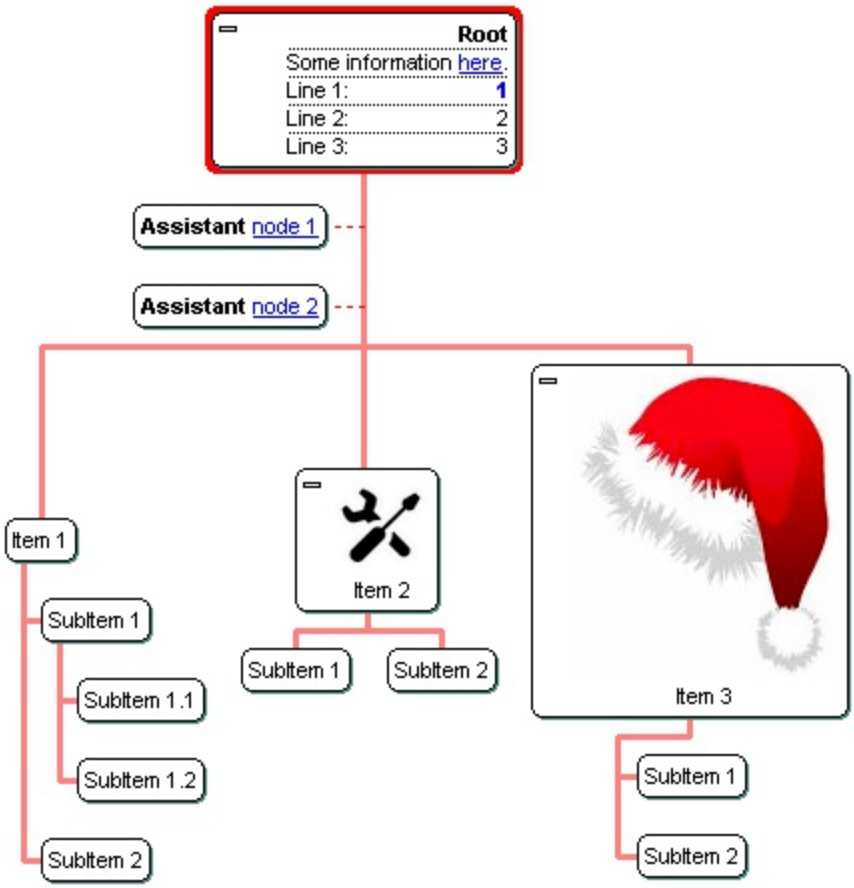
Type	Description
Boolean	A Boolean expression that specifies whether links are shown rounded or rectangular.

By default, the ShowRoundLink property is False. Use the ShowRoundLink property to specify round links for all nodes. Use the ShowRoundLink property to specify round links for a specified node and its child nodes. Use the PenLink property indicates the type of the pen used to paint the links between nodes. Use the PenWidthLink property to specify the thickness of the links between nodes. Use the LinkColor property to specify the color for the links between nodes. The LinkToRound property specifies the whether the link is shown liner or round .

The following screen shot shows the chart using round links (ShowRoundLink property is True) :



The following screen shot shows the chart using round links (ShowRoundLink property is False) :


Root
Some information here.
Line 1: 1
Line 2: 2
Line 3: 3
Assistant node 1
Assistant node 2
Item 1
SubItem 1
SubItem 1.1
SubItem 1.2
SubItem 2
Item 2
SubItem 1
SubItem 2
Item 3
SubItem 1
SubItem 2

method ChartView.ShowToolTip (ToolTip as String, [Title as Variant], [Alignment as Variant], [X as Variant], [Y as Variant])

Shows the specified tooltip at given position.

Type	Description
ToolTip as String	The ToolTip parameter can be any of the following: • NULL(BSTR) or "<null>"(string) to indicate that the tooltip for the object being hovered is not changed • A String expression that indicates the description of the tooltip, that supports built-in HTML format (adds, replaces or changes the object's tooltip)
Title as Variant	The Title parameter can be any of the following: • missing (VT_EMPTY, VT_ERROR type) or "<null>" (string) the title for the object being hovered is not changed. • A String expression that indicates the title of the tooltip (no built-in HTML format) (adds, replaces or changes the object's title)
Alignment as Variant	A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing (VT_EMPTY, VT_ERROR) the alignment of the tooltip for the object being hovered is not changed. The Alignment parameter can be one of the following: • 0 - exTopLeft • 1 - exTopRight • 2 - exBottomLeft • 3 - exBottomRight • 0x10 - exCenter • 0x11 - exCenterLeft • 0x12 - exCenterRight • 0x13 - exCenterTop • 0x14 - exCenterBottom By default, the tooltip is aligned relative to the top-left corner (0 - exTopLeft).

X as Variant	Specifies the horizontal position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current horizontal position of the cursor (current x-position) • a numeric expression that indicates the horizontal screen position to show the tooltip (fixed screen x-position) • a string expression that indicates the horizontal displacement relative to default position to show the tooltip (moved)
Y as Variant	Specifies the vertical position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current vertical position of the cursor (current y-position) • a numeric expression that indicates the vertical screen position to show the tooltip (fixed screen y-position) • a string expression that indicates the vertical displacement relative to default position to show the tooltip (displacement)

Use the ShowToolTip method to display a custom tooltip at specified position or to update the object's tooltip, title or position. You can call the ShowToolTip method during the MouseMove event. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

For instance:

- ShowToolTip(`<null>`,`<null>`,,`+8`,`+8`), shows the tooltip of the object moved relative

to its default position

- ShowToolTip(`<null>`,`new title`), adds, changes or replaces the title of the object's tooltip
- ShowToolTip(`new content`), adds, changes or replaces the object's tooltip
- ShowToolTip(`new content`,`new title`), shows the tooltip and title at current position
- ShowToolTip(`new content`,`new title`,,`+8`,`+8`), shows the tooltip and title moved relative to the current position
- ShowToolTip(`new content`,``,,128,128), displays the tooltip at a fixed position
- ShowToolTip(``,``), hides the tooltip

The ToolTip parameter supports the built-in HTML format like follows:

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

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

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

  gradient-center

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

  outlined

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

  shadow

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

  outline anti-aliasing

property ChartView.Template as String

Specifies the control's template.

Type	Description
String	A String expression that indicates the control's template

The control's template uses the X-Script language to initialize the control's content. Use the Template property page of the control to update the control's Template property. Use the Template property to execute code by passing instructions as a string (template string). The ExecuteTemplate property gets the result of executing a template script.

Most of our UI components provide a Template page that's accessible in design mode. No matter what programming language you are using, you can have a quick view of the component's features using the WYSWYG Template editor.

- Place the control to your form or dialog.
- Locate the Properties item, in the control's context menu, in design mode. If your environment doesn't provide a Properties item in the control's context menu, please try to locate in the Properties browser.
- Click it, and locate the Template page.
- Click the Help button. In the left side, you will see the component, in the right side, you will see a x-script code that calls methods and properties of the control.

The control's Template page helps user to initialize the control's look and feel in design mode, using the x-script language that's easy and powerful. The Template page displays the control on the left side of the page. On the right side of the Template page, a simple editor is displayed where user writes the initialization code. The control's look and feel is automatically updated as soon as the user types new instructions. The Template script is saved to the container persistence (when Apply button is pressed), and it is executed when the control is initialized at runtime. Any component that provides a WYSWYG Template page, provides a Template property. The Template property executes code from a string (template string).

The Template script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters.

An instruction can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values*

separated by commas. (Sample: h = InsertItem(0,"New Child"))

- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments).... *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The Template supports the following general functions:

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier*

property ChartView.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 ChartView.TemplatePut (NewVal as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
NewVal as Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplatePut method / TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters) = Value*, so our controls provide an alternative using the TemplateDef / TemplatePut method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

The TemplateDef, TemplatePut, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments).... *The .(dot) character splits the object from its property. For instance, the*

Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property ChartView.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.

By default, the ToolTipDelay property is 500 ms. If the ToolTipDelay property is 0, the control displays no tooltips for any keyword. Use the ToolTipDelay and ToolTipPopDelay properties to define the time before showing a tooltip and the period of the time that tooltip remains visible if the mouse pointer is stationary within a control. The ToolTipWidth property specifies a value that indicates the width of the tooltip window, in pixels. 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 ToolTipFont property to assign a font for the control's tooltip. Use the ToolTip property to assign a tooltip to a node.

property ChartView.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. Use the ToolTipDelay and ToolTipPopDelay properties to define the time before showing a tooltip and the period of the time that tooltip remains visible if the mouse pointer is stationary within a control. The ToolTipWidth property specifies a value that indicates the width of the tooltip window, in pixels. Use the ToolTip property to assign a tooltip to a node. Use the ToolTipFont property to assign a font for the control's tooltip. You can use the <*font*> HTML element, in the tooltip's description to assign a different font for portions of text.

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

By default, the ToolTipPopDelay property is 5000 ms. If the ToolTipPopDelay property is 0, the control displays no tooltips for any keyword. Use the ToolTipDelay and ToolTipPopDelay properties to define the time before showing a tooltip and the period of the time that tooltip remains visible if the mouse pointer is stationary within a control. The ToolTipWidth property specifies a value that indicates the width of the tooltip window, in pixels. Use the ToolTipFont property to assign a font for the control's tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip. Use the ToolTip property to assign a tooltip to a node.

property ChartView.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 specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. 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 Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip. Use the ToolTipFont property to assign a font for the control's tooltip. Use the ToolTip property to assign a tooltip to a node.

property ChartView.ToTemplate ([DefaultTemplate as Variant]) as String

Generates the control's template.

Type	Description
DefaultTemplate as Variant	A String expression that indicates the default format used to define the control's template at runtime, or a string expression that indicates the path to the file being used to define the default template (like c:\temp\teml.bin). If it is missing (by default), the control's uses the default implementation (listed bellow) to define the control's template, at runtime. Each line in the DefaultTemplate parameter, defines a property or an instruction to generate the template.
String	A String expression that indicates the control's template.

Use the ToTemplate property to save the control's content to a template string. The ToTemplate property saves the control's properties based on the default template. Use the ToTemplate property to copy the control's content to another instance. The ToTemplate property can save pictures, icons, binary arrays, objects, collections, and so on based on the DefaultTemplate parameter.

The DefaultTemplate parameter indicates the format of the template being used to generate the control's template at runtime. If the DefaultTemplate parameter is missing, the control's uses its default template listed bellow. The DefaultTemplate parameter defines the list of properties and instructions that generates the control's template. Remove the properties and objects, in the default template, that you don't need in the generated template script. Use the Template property to apply the template to the control. Use the Template property to execute code by passing instructions as a string (template string). The Template script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters. The Template format contains a list of instructions that loads data and change properties for the objects in the control. Use the AllowCopyTemplate property to copy the control's content to the clipboard, in template format, using the the Shift + Ctrl + Alt + Insert sequence.

The time to generate the control's template depends on:

- the content of the DefaultTemplate parameter.
- number of columns and items in the control including internal objects such as editors.
- encoding the visual appearance as well as encoding the pictures and icons of the control

property ChartView.Version as String

Retrieves the control's version.

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

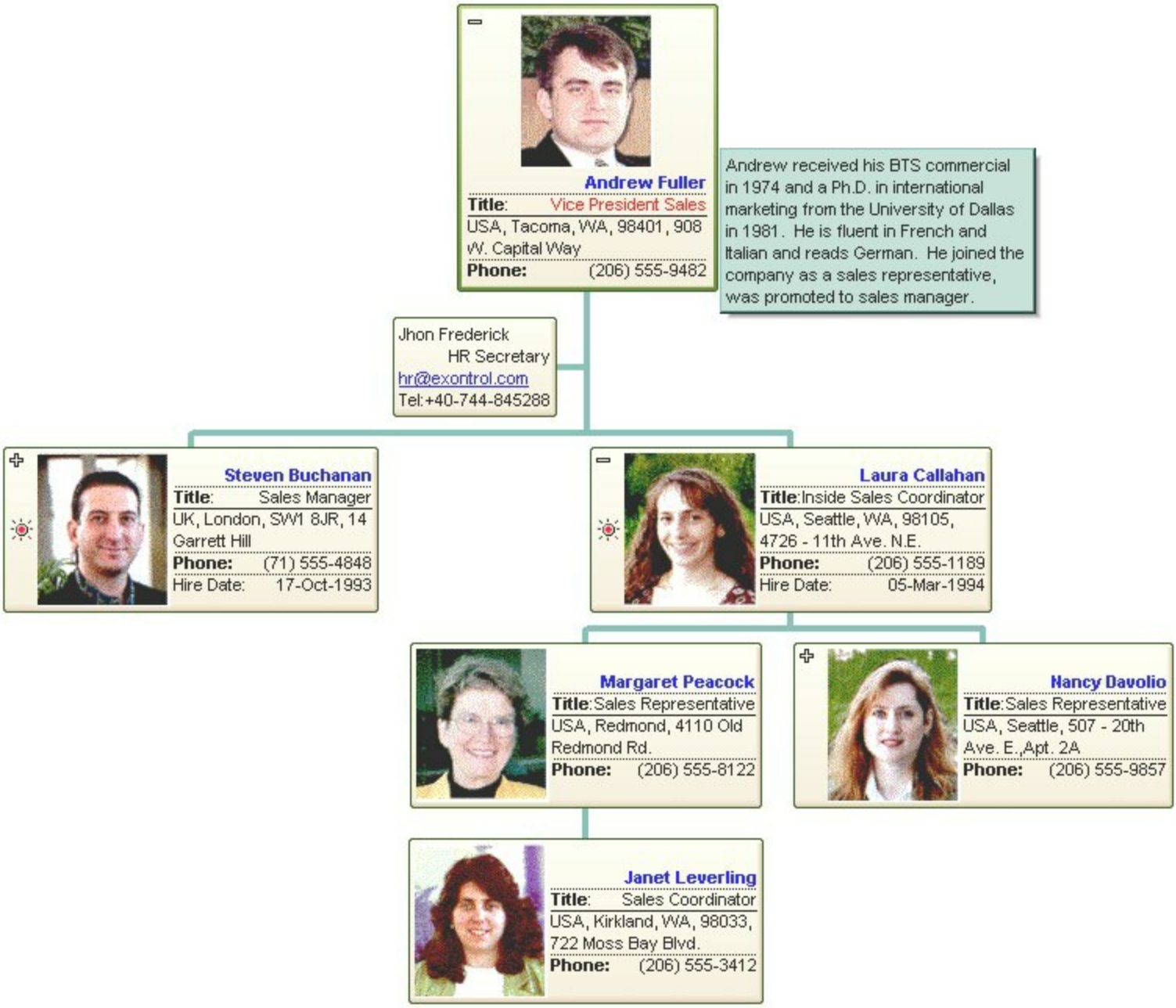
The Version property to indicates the version of the control that's running.

property ChartView.VisualAppearance as Appearance

Retrieves the control's appearance.

Type	Description
Appearance	An Appearance object that holds a collection of skins.

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part.



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

- selected node, SelColor property

- borders of the nodes, Background property
- the background for all nodes, BackColorNode property
- node's background, BackColor property

property ChartView.WidthNode as Long

Specifies the maximum width of the nodes captions.

Type	Description
Long	A Long expression that specifies a maximum width for all nodes, when their captions are wrapped.

Use the NodeWidth property to specify the maximum width for all nodes, when their captions are wrapped. Use the Width property to specify the maximum width a specified node when it's caption is wrapped.

property ChartView.ZoomHeight as Double

Gets or sets a value indicating how large the chart will appear on vertical axis.

Type	Description
Double	A double expression that indicates how large the chart will appear on vertical axis.

Use the ZoomHeight property to change how large the chart will appear on vertical axis, if the ZoomHeightMode property is exCustomSize. Use the ZoomWidth property to specify how large the chart will appear on horizontal axis. Use the MinZoomHeight/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking.

The following VB sample enlarges the chart to 200%:

```
With ChartView1
    .BeginUpdate
      .ZoomWidthMode = exCustomSize
      .ZoomWidth = 2
      .ZoomHeightMode = exCustomSize
      .ZoomHeight = 2
    .EndUpdate
End With
```

The following C++ sample enlarges the chart to 200%:

```
m_chartview.BeginUpdate();
m_chartview.SetZoomWidthMode( 1 /*exCustomSize*/ );
m_chartview.SetZoomWidth( 2 );
m_chartview.SetZoomHeightMode( 1 /*exCustomSize*/ );
m_chartview.SetZoomHeight( 2 );
m_chartview.EndUpdate();
```

The following VB.NET sample enlarges the chart to 200%:

```
With AxChartView1
  .BeginUpdate()
  .ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize
  .ZoomWidth = 2
```

```
    .ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize
    .ZoomHeight = 2
    .EndUpdate()
End With
```

The following C# sample enlarges the chart to 200%:

```
axChartView1.BeginUpdate();
axChartView1.ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize;
axChartView1.ZoomWidth = 2;
axChartView1.ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize;
axChartView1.ZoomHeight = 2;
axChartView1.EndUpdate();
```

The following VFP sample enlarges the chart to 200%:

```
With thisform.ChartView1
    .BeginUpdate
    .ZoomWidthMode = 1 && exCustomSize
    .ZoomWidth = 2
    .ZoomHeightMode = 1 && exCustomSize
    .ZoomHeight = 2
    .EndUpdate
EndWith
```

property ChartView.ZoomHeightMode as ZoomModeEnum

Specifies a value that indicates whether the ZoomHeight property is updated when the control is resized.

Type	Description
ZoomModeEnum	A ZoomModeEnum expression that indicates whether the ZoomHeight property is updated when the control is resized.

By default, the ZoomHeightMode property is exDefaultSize. If the ZoomHeightMode property is exCustomSize the ZoomHeight property specifies how large the chart will appear on the vertical axis. If the ZoomHeightMode property is exControlSize the control updates the ZoomHeight property such that the chart will fit the control's client are on vertical axis. Use the ZoomWidthMode property to specify whether the ZoomWidth property is updated when the control is resized.

The following VB sample fits the chart to the control's client area:

```
With ChartView1
    .BeginUpdate
      .ZoomWidthMode = exControlSize
      .ZoomHeightMode = exControlSize
    .EndUpdate
End With
```

The following C++ sample fits the chart to the control's client area:

```
m_chartview.BeginUpdate();
m_chartview.SetZoomWidthMode( 2 /*exControlSize*/ );
m_chartview.SetZoomHeightMode( 2 /*exControlSize*/ );
m_chartview.EndUpdate();
```

The following VB.NET sample fits the chart to the control's client area:

```
With AxChartView1
  .BeginUpdate()
  .ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exControlSize
  .ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exControlSize
  .EndUpdate()
End With
```

The following C# sample fits the chart to the control's client area:

```
axChartView1.BeginUpdate();
axChartView1.ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exControlSize;
axChartView1.ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exControlSize;
axChartView1.EndUpdate();
```

The following VFP sample fits the chart to the control's client area:

```
With thisform.ChartView1
   .BeginUpdate
   .ZoomWidthMode = 2 && exControlSize
   .ZoomHeightMode = 2 && exControlSize
   .EndUpdate
EndWith
```

property ChartView.ZoomWidth as Double

Gets or sets a value indicating how large the chart will appear on horizontal axis.

Type	Description
Double	A double expression that indicates how large the chart will appear on horizontal axis.

Use the ZoomWidth property to change how large the chart will appear on horizontal axis, if the ZoomWidthMode property is exCustomSize. Use the ZoomHeight property to specify how large the chart will appear on vertical axis. Use the MinZoomWidth/MaxZoomWidth property to specify the limits on horizontal axis when the user performs resizing/zooming/shrinking.

The following VB sample enlarges the chart to 200%:

```
With ChartView1
    .BeginUpdate
      .ZoomWidthMode = exCustomSize
      .ZoomWidth = 2
      .ZoomHeightMode = exCustomSize
      .ZoomHeight = 2
    .EndUpdate
End With
```

The following C++ sample enlarges the chart to 200%:

```
m_chartview.BeginUpdate();
m_chartview.SetZoomWidthMode( 1 /*exCustomSize*/ );
m_chartview.SetZoomWidth( 2 );
m_chartview.SetZoomHeightMode( 1 /*exCustomSize*/ );
m_chartview.SetZoomHeight( 2 );
m_chartview.EndUpdate();
```

The following VB.NET sample enlarges the chart to 200%:

```
With AxChartView1
  .BeginUpdate()
  .ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize
  .ZoomWidth = 2
```

```
    .ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize
    .ZoomHeight = 2
    .EndUpdate()
End With
```

The following C# sample enlarges the chart to 200%:

```
axChartView1.BeginUpdate();
axChartView1.ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize;
axChartView1.ZoomWidth = 2;
axChartView1.ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exCustomSize;
axChartView1.ZoomHeight = 2;
axChartView1.EndUpdate();
```

The following VFP sample enlarges the chart to 200%:

```
With thisform.ChartView1
    .BeginUpdate
    .ZoomWidthMode = 1 && exCustomSize
    .ZoomWidth = 2
    .ZoomHeightMode = 1 && exCustomSize
    .ZoomHeight = 2
    .EndUpdate
EndWith
```

property ChartView.ZoomWidthMode as ZoomModeEnum

Specifies a value that indicates whether the ZoomWidth property is updated when the control is resized.

Type	Description
ZoomModeEnum	A ZoomModeEnum expression that indicates whether the ZoomWidth property is updated when the control is resized.

By default, the ZoomWidthMode property is exDefaultSize. If the ZoomWidthMode property is exCustomSize the ZoomWidth property specifies how large the chart will appear on the horizontal axis. If the ZoomWidthMode property is exControlSize the control updates the ZoomWidth property such that the chart will fit the control's client are on horizontal axis. Use the ZoomHeightMode property to specify whether the ZoomHeight property is updated when the control is resized.

The following VB sample fits the chart to the control's client area:

```
With ChartView1
    .BeginUpdate
      .ZoomWidthMode = exControlSize
      .ZoomHeightMode = exControlSize
    .EndUpdate
End With
```

The following C++ sample fits the chart to the control's client area:

```
m_chartview.BeginUpdate();
m_chartview.SetZoomWidthMode( 2 /*exControlSize*/ );
m_chartview.SetZoomHeightMode( 2 /*exControlSize*/ );
m_chartview.EndUpdate();
```

The following VB.NET sample fits the chart to the control's client area:

```
With AxChartView1
  .BeginUpdate()
  .ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exControlSize
  .ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exControlSize
  .EndUpdate()
End With
```

The following C# sample fits the chart to the control's client area:

```
axChartView1.BeginUpdate();
axChartView1.ZoomWidthMode = EXORGCHARTLib.ZoomModeEnum.exControlSize;
axChartView1.ZoomHeightMode = EXORGCHARTLib.ZoomModeEnum.exControlSize;
axChartView1.EndUpdate();
```

The following VFP sample fits the chart to the control's client area:

```
With thisform.ChartView1
   .BeginUpdate
   .ZoomWidthMode = 2 && exControlSize
   .ZoomHeightMode = 2 && exControlSize
   .EndUpdate
EndWith
```

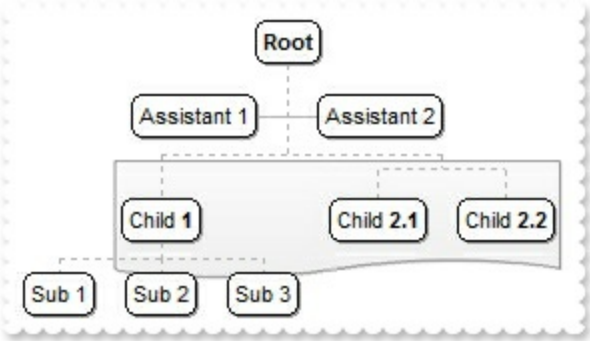
Frame object

The Frame objects holds information about what a frame to display. The Add method returns the newly added frame. The Item property gets a frame object based on its index.

A Frame is defined by an union of nodes, and can:

- specify whether the frame is shown on the back or on the front using the ShowOnBackground property
- define the padding of the frame using the Padding property
- define a solid or EBN background color to be displayed on the frame's background, using the BackColor property of the Frame object
- The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background.
- define a different border or pattern to be shown, using the Pattern property

The following screen show shows an EBN frame around child nodes of the root node:



The Frame object supports the following properties and methods:

Name	Description
BackColor	Gets or sets a value that indicates the frame's background color.
BackgroundExt	Indicates additional colors, text, images that can be displayed on the frame's background using the EBN string format.
BackgroundExtValue	Specifies at runtime, the value of the giving property for specified part of the background extension.
Index	Indicates the index of the current Frame object within the Frames collection.
Nodes	Specifies the list of keys for the nodes to show the frame, separated by comma character. The node's key may ends with (all) to include all child, assistant nodes, with (child) to

	include the direct children of the node only, with (assistant) to include the as?ł?K
Padding	Returns or sets a value that indicates the padding of the frame.
Pattern	Specifies the pattern of the frame.
ShowOnBackground	Shows the frame on the control's background.
Visible	Shows or hides the frame.

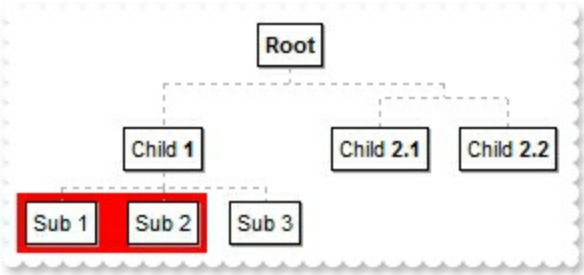
property Frame.BackColor as Color

Gets or sets a value that indicates the frame's background color.

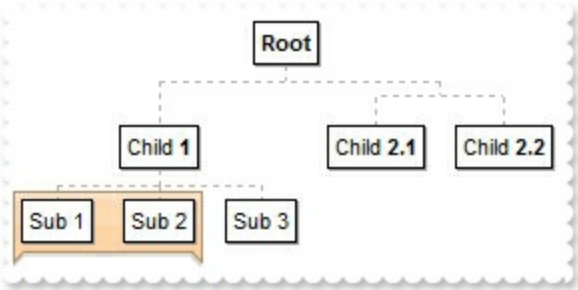
Type	Description
Color	A color expression that defines the frame's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part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

By default, the BackColor property is -1, which indicates that no background color is applied. The BackColor property defines the frame's background color. The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background. The Pattern property defines a different border or pattern to be shown. The Padding property returns or sets a value that indicates the padding of the frame.

The following screen shot shows the frame with a solid background color:



The following screen shot shows the frame with an EBN background color:



The following screen shot shows how you can define a solid background color for the frame (with no border).

VBA (MS Access, Excell...)

```
With ChartView1
```

```
    .BeginUpdate
    With .Nodes
        .Add "Child <b>1</b>",,"1234"
        .Add "Sub 1","1234","AK1"
        .Add "Sub 2","1234","AK2"
        .Add "Sub 3","1234"
        .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
    End With
    With .Frames.Add("AK1,AK2")
        .BackColor = RGB(255,0,0)
        .Pattern.Type = 0
    End With
    .EndUpdate
End With
```

VB6

```
With ChartView1
    .BeginUpdate
    With .Nodes
        .Add "Child <b>1</b>",,"1234"
        .Add "Sub 1","1234","AK1"
        .Add "Sub 2","1234","AK2"
        .Add "Sub 3","1234"
        .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
    End With
    With .Frames.Add("AK1,AK2")
        .BackColor = RGB(255,0,0)
        .Pattern.Type = exPatternEmpty
    End With
    .EndUpdate
End With
```

VB.NET

```
With Exchartview1
    .BeginUpdate()
    With .Nodes
```

```
        .Add("Child <b>1</b>",,"1234")
        .Add("Sub 1","1234","AK1")
        .Add("Sub 2","1234","AK2")
        .Add("Sub 3","1234")
        .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
    End With
    With .Frames.Add("AK1,AK2")
        .BackColor = Color.FromArgb(255,0,0)
        .Pattern.Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternEmpty
    End With
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxChartView1
    .BeginUpdate()
    With .Nodes
        .Add("Child <b>1</b>",,"1234")
        .Add("Sub 1","1234","AK1")
        .Add("Sub 2","1234","AK2")
        .Add("Sub 3","1234")
        .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
    End With
    With .Frames.Add("AK1,AK2")
        .BackColor = RGB(255,0,0)
        .Pattern.Type = EXORGCHARTLib.PatternEnum.exPatternEmpty
    End With
    .EndUpdate()
End With
```

C++

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

```
    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->BeginUpdate();
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"Child <b>1</b>",vtMissing,"1234",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 1","1234","AK1",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 2","1234","AK2",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 3","1234",vtMissing,vtMissing,vtMissing);
   var_Nodes->Add(L"Child <b>2.1</b>",vtMissing,vtMissing,vtMissing,vtMissing)-
>AddGroup(L"Child <b>2.2</b>",vtMissing,vtMissing);
EXORGCHARTLib::IFramePtr var_Frame = spChartView1->GetFrames()-
>Add("AK1,AK2");
   var_Frame->PutBackColor(RGB(255,0,0));
   var_Frame->GetPattern()->PutType(EXORGCHARTLib::exPatternEmpty);
spChartView1->EndUpdate();
```

C++ Builder

```
ChartView1->BeginUpdate();
Exorgchartlib_tlb::INodesPtr var_Nodes = ChartView1->Nodes;
   var_Nodes->Add(L"Child
<b>1</b>",TNoParam(),TVariant("1234"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub
1",TVariant("1234"),TVariant("AK1"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub
2",TVariant("1234"),TVariant("AK2"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub 3",TVariant("1234"),TNoParam(),TNoParam(),TNoParam());
   var_Nodes->Add(L"Child
<b>2.1</b>",TNoParam(),TNoParam(),TNoParam(),TNoParam())->AddGroup(L"Child
<b>2.2</b>",TNoParam(),TNoParam());
Exorgchartlib_tlb::IFramePtr var_Frame = ChartView1->Frames-
>Add(TVariant("AK1,AK2"));
   var_Frame->BackColor = RGB(255,0,0);
```

```
   var_Frame->Pattern->Type = Exorgchartlib_tlb::PatternEnum::exPatternEmpty;
ChartView1->EndUpdate();
```

C#

```
exchartview1.BeginUpdate();
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
   var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
   var_Nodes.Add("Sub 1","1234","AK1",null,null);
   var_Nodes.Add("Sub 2","1234","AK2",null,null);
   var_Nodes.Add("Sub 3","1234",null,null,null);
   var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
exontrol.EXORGCHARTLib.Frame var_Frame = exchartview1.Frames.Add("AK1,AK2");
   var_Frame.BackColor = Color.FromArgb(255,0,0);
   var_Frame.Pattern.Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternEmpty;
exchartview1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   ChartView1.BeginUpdate();
   var var_Nodes = ChartView1.Nodes;
      var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
      var_Nodes.Add("Sub 1","1234","AK1",null,null);
      var_Nodes.Add("Sub 2","1234","AK2",null,null);
      var_Nodes.Add("Sub 3","1234",null,null,null);
      var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
   var var_Frame = ChartView1.Frames.Add("AK1,AK2");
```

```
        var_Frame.BackColor = 255;
        var_Frame.Pattern.Type = 0;
    ChartView1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With ChartView1
        .BeginUpdate
        With .Nodes
            .Add "Child <b>1</b>",,"1234"
            .Add "Sub 1","1234","AK1"
            .Add "Sub 2","1234","AK2"
            .Add "Sub 3","1234"
            .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
        End With
        With .Frames.Add("AK1,AK2")
            .BackColor = RGB(255,0,0)
            .Pattern.Type = 0
        End With
        .EndUpdate
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axChartView1.BeginUpdate();
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
   var_Nodes.Add("Sub 1","1234","AK1",null,null);
   var_Nodes.Add("Sub 2","1234","AK2",null,null);
   var_Nodes.Add("Sub 3","1234",null,null,null);
   var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
EXORGCHARTLib.Frame var_Frame = axChartView1.Frames.Add("AK1,AK2");
   var_Frame.BackColor = (uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0));
   var_Frame.Pattern.Type = EXORGCHARTLib.PatternEnum.exPatternEmpty;
axChartView1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Frame,com_Node,com_Nodes,com_Pattern;
   anytype var_Frame,var_Node,var_Nodes,var_Pattern;
   ;

   super();

   exchartview1.BeginUpdate();
   var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
      com_Nodes.Add("Child <b>1</b>",,"1234");
      com_Nodes.Add("Sub 1","1234","AK1");
      com_Nodes.Add("Sub 2","1234","AK2");
      com_Nodes.Add("Sub 3","1234");
      var_Node = COM::createFromObject(com_Nodes.Add("Child <b>2.1</b>"));
com_Node = var_Node;
      com_Node.AddGroup("Child <b>2.2</b>");
   var_Frame = COM::createFromObject(exchartview1.Frames()).Add("AK1,AK2");
com_Frame = var_Frame;
      com_Frame.BackColor(WinApi::RGB2int(255,0,0));
      var_Pattern = COM::createFromObject(com_Frame.Pattern()); com_Pattern =
```

```
var_Pattern;
      com_Pattern.Type(0/*exPatternEmpty*/);
   exchartview1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
   BeginUpdate();
   with Nodes do
   begin
      Add('Child <b>1</b>',Nil,'1234',Nil,Nil);
      Add('Sub 1','1234','AK1',Nil,Nil);
      Add('Sub 2','1234','AK2',Nil,Nil);
      Add('Sub 3','1234',Nil,Nil,Nil);
      Add('Child <b>2.1</b>',Nil,Nil,Nil,Nil).AddGroup('Child <b>2.2</b>',Nil,Nil);
   end;
   with Frames.Add('AK1,AK2') do
   begin
      BackColor := $ff;
      Pattern.Type := EXORGCHARTLib.PatternEnum.exPatternEmpty;
   end;
   EndUpdate();
end
```

Delphi (standard)

```
with ChartView1 do
begin
   BeginUpdate();
   with Nodes do
   begin
      Add('Child <b>1</b>',Null,'1234',Null,Null);
      Add('Sub 1','1234','AK1',Null,Null);
      Add('Sub 2','1234','AK2',Null,Null);
      Add('Sub 3','1234',Null,Null,Null);
      Add('Child <b>2.1</b>',Null,Null,Null,Null).AddGroup('Child
```

```
<b>2.2</b>',Null,Null);
  end;
  with Frames.Add('AK1,AK2') do
  begin
    BackColor := $ff;
    Pattern.Type := EXORGCHARTLib_TLB.exPatternEmpty;
  end;
  EndUpdate();
end
```

VFP

```
with thisform.ChartView1
  .BeginUpdate
  with .Nodes
    .Add("Child <b>1</b>",Null,"1234")
    .Add("Sub 1","1234","AK1")
    .Add("Sub 2","1234","AK2")
    .Add("Sub 3","1234")
    .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
  endwith
  with .Frames.Add("AK1,AK2")
    .BackColor = RGB(255,0,0)
    .Pattern.Type = 0
  endwith
  .EndUpdate
endwith
```

dBASE Plus

```
local oChartView,var_Frame,var_Nodes

oChartView = form.EXORGCHARTACTIVEXCONTROL1.nativeObject
oChartView.BeginUpdate()
var_Nodes = oChartView.Nodes
  var_Nodes.Add("Child <b>1</b>",null,"1234")
  var_Nodes.Add("Sub 1","1234","AK1")
  var_Nodes.Add("Sub 2","1234","AK2")
```

```
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Frame = oChartView.Frames.Add("AK1,AK2")
   var_Frame.BackColor = 0xff
   var_Frame.Pattern.Type = 0
oChartView.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oChartView as P
Dim var_Frame as P
Dim var_Nodes as P

oChartView = topparent:CONTROL_ACTIVEX1.activex
oChartView.BeginUpdate()
var_Nodes = oChartView.Nodes
   var_Nodes.Add("Child <b>1</b>",,"1234")
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Frame = oChartView.Frames.Add("AK1,AK2")
   var_Frame.BackColor = 255
   var_Frame.Pattern.Type = 0
oChartView.EndUpdate()
```

Visual Objects

```
local var_Frame as IFrame
local var_Nodes as INodes

oDCOCX_Exontrol1:BeginUpdate()
var_Nodes := oDCOCX_Exontrol1:Nodes
   var_Nodes:Add("Child <b>1</b>",nil,"1234",nil,nil)
   var_Nodes:Add("Sub 1","1234","AK1",nil,nil)
   var_Nodes:Add("Sub 2","1234","AK2",nil,nil)
```

```
   var_Nodes:Add("Sub 3","1234",nil,nil,nil)
   var_Nodes:Add("Child <b>2.1</b>",nil,nil,nil,nil):AddGroup("Child
<b>2.2</b>",nil,nil)
var_Frame := oDCOCX_Exontrol1:Frames:Add("AK1,AK2")
   var_Frame:BackColor := RGB(255,0,0)
   var_Frame:Pattern:Type := exPatternEmpty
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oChartView,var_Frame,var_Nodes

oChartView = ole_1.Object
oChartView.BeginUpdate()
var_Nodes = oChartView.Nodes
   var_Nodes.Add("Child <b>1</b>",,"1234")
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Frame = oChartView.Frames.Add("AK1,AK2")
   var_Frame.BackColor = RGB(255,0,0)
   var_Frame.Pattern.Type = 0
oChartView.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Variant voNodes
   Get ComNodes to voNodes
   Handle hoNodes
   Get Create (RefClass(cComNodes)) to hoNodes
   Set pvComObject of hoNodes to voNodes
      Get ComAdd of hoNodes "Child <b>1</b>" "1234" Nothing Nothing to
```

```
Nothing
      Get ComAdd of hoNodes "Sub 1" "1234" "AK1" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "Sub 2" "1234" "AK2" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "Sub 3" "1234" Nothing Nothing Nothing to Nothing
      Variant voNode
      Get ComAdd of hoNodes "Child <b>2.1</b>" Nothing Nothing Nothing
Nothing to voNode
      Handle hoNode
      Get Create (RefClass(cComNode)) to hoNode
      Set pvComObject of hoNode to voNode
         Get ComAddGroup of hoNode "Child <b>2.2</b>" Nothing Nothing to
Nothing
      Send Destroy to hoNode
   Send Destroy to hoNodes
   Variant voFrames
   Get ComFrames to voFrames
   Handle hoFrames
   Get Create (RefClass(cComFrames)) to hoFrames
   Set pvComObject of hoFrames to voFrames
      Variant voFrame
      Get ComAdd of hoFrames "AK1,AK2" to voFrame
      Handle hoFrame
      Get Create (RefClass(cComFrame)) to hoFrame
      Set pvComObject of hoFrame to voFrame
         Set ComBackColor of hoFrame to (RGB(255,0,0))
         Variant voPattern
         Get ComPattern of hoFrame to voPattern
         Handle hoPattern
         Get Create (RefClass(cComPattern)) to hoPattern
         Set pvComObject of hoPattern to voPattern
            Set ComType of hoPattern to OLEexPatternEmpty
         Send Destroy to hoPattern
      Send Destroy to hoFrame
   Send Destroy to hoFrames
   Send ComEndUpdate
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oChartView
   LOCAL oFrame
   LOCAL oNodes

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

   oChartView := XbpActiveXControl():new( oForm:drawingArea )
   oChartView:CLSID  := "Exontrol.ChartView.1" /*{F4DFE455-01FE-420E-A088-
64346DCC3791}*/
   oChartView:create(,, {10,60},{610,370} )

      oChartView:BeginUpdate()
      oNodes := oChartView:Nodes()
         oNodes:Add("Child <b>1</b>",,"1234")
         oNodes:Add("Sub 1","1234","AK1")
         oNodes:Add("Sub 2","1234","AK2")
         oNodes:Add("Sub 3","1234")
         oNodes:Add("Child <b>2.1</b>"):AddGroup("Child <b>2.2</b>")
      oFrame := oChartView:Frames():Add("AK1,AK2")
         oFrame:SetProperty("BackColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 255,0,0 } )  , .F. ))
         oFrame:Pattern():Type := 0/*exPatternEmpty*/
      oChartView:EndUpdate()

   oForm:Show()
   DO WHILE nEvent != xbeP_Quit
```

```
      nEvent := AppEvent( @mp1, @mp2, @oXbp )
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO
RETURN
```

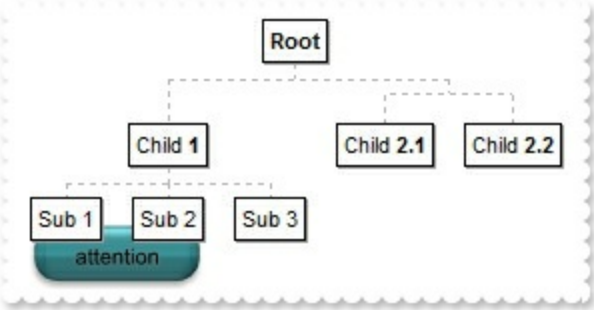
property Frame.BackgroundExt as String

Indicates additional colors, text, images that can be displayed on the frame's background using the EBN string format.

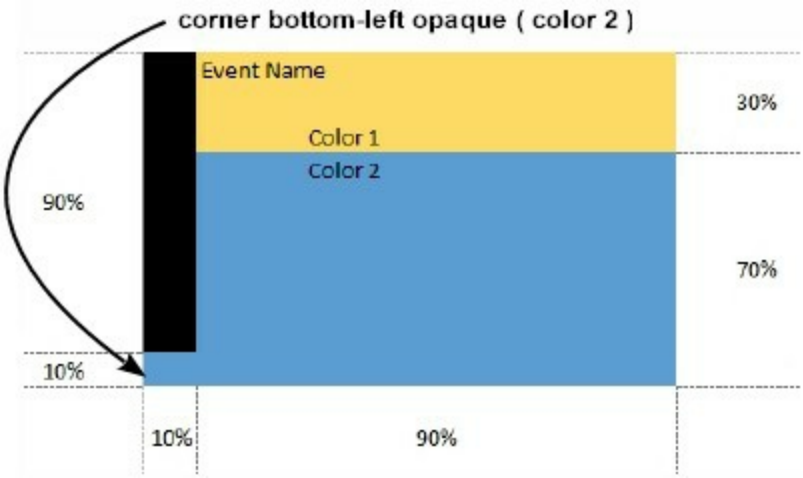
Type	Description
String	A String expression ("**EBN String Format**") that defines the layout of the UI to be applied on the object's background. The syntax of EBN String Format in BNF notation is shown bellow. *You can use the EBN's Builder of eXButton/COM control to define visually the EBN String Format.*

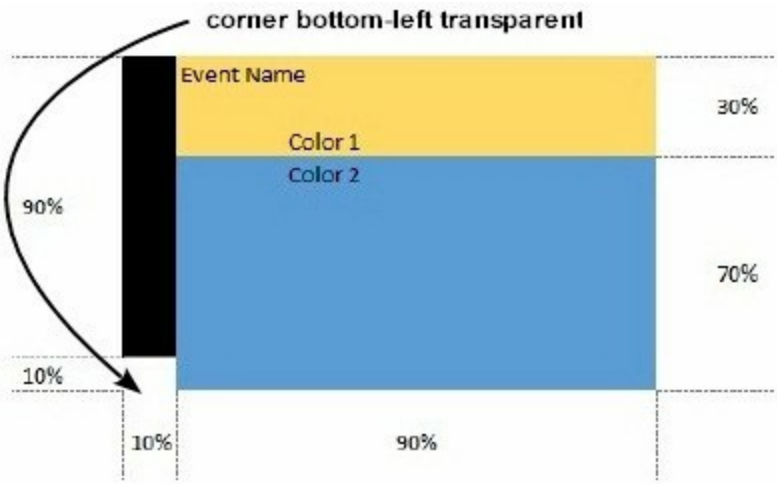
By default, the BackgroundExt property is "", which indicates that no background-extension is applied. The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background. Using the BackgroundExt property you have unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background. *For instance, let's say you need to display **more** colors on the frame's background, or just want to display an **additional** caption or image to a specified location on the object's background.* The EBN String Format defines the parts of the EBN to be applied on the object's background. The EBN is a set of UI elements that are built as a tree where each element is anchored to its parent element. Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExt property is applied right after setting the object's backcolor, and before drawing the default object's captions, icons or pictures. The BackColor property defines the frame's background color. The Pattern property defines a different border or pattern to be shown. The Padding property returns or sets a value that indicates the padding of the frame.

The following screen shot shows the frame with a caption on an EBN object:



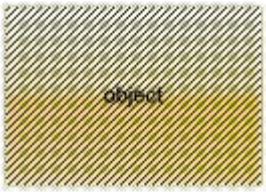
Complex samples:



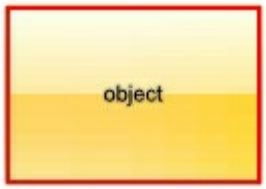


Easy samples:

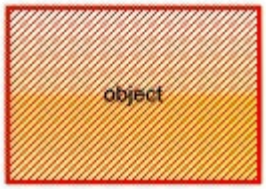
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



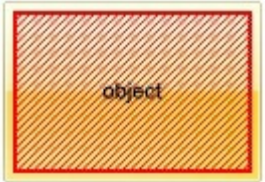
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



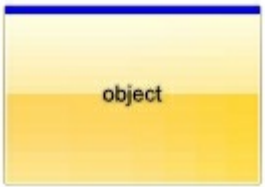
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



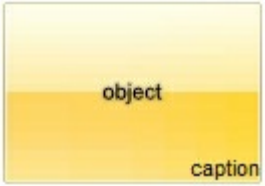
- "[[patterncolor=RGB(255,0,0)] (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.

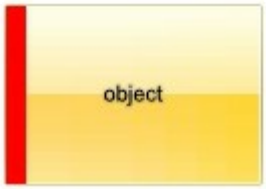


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

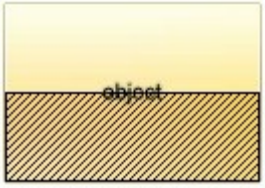


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

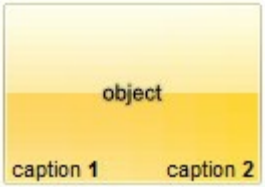
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



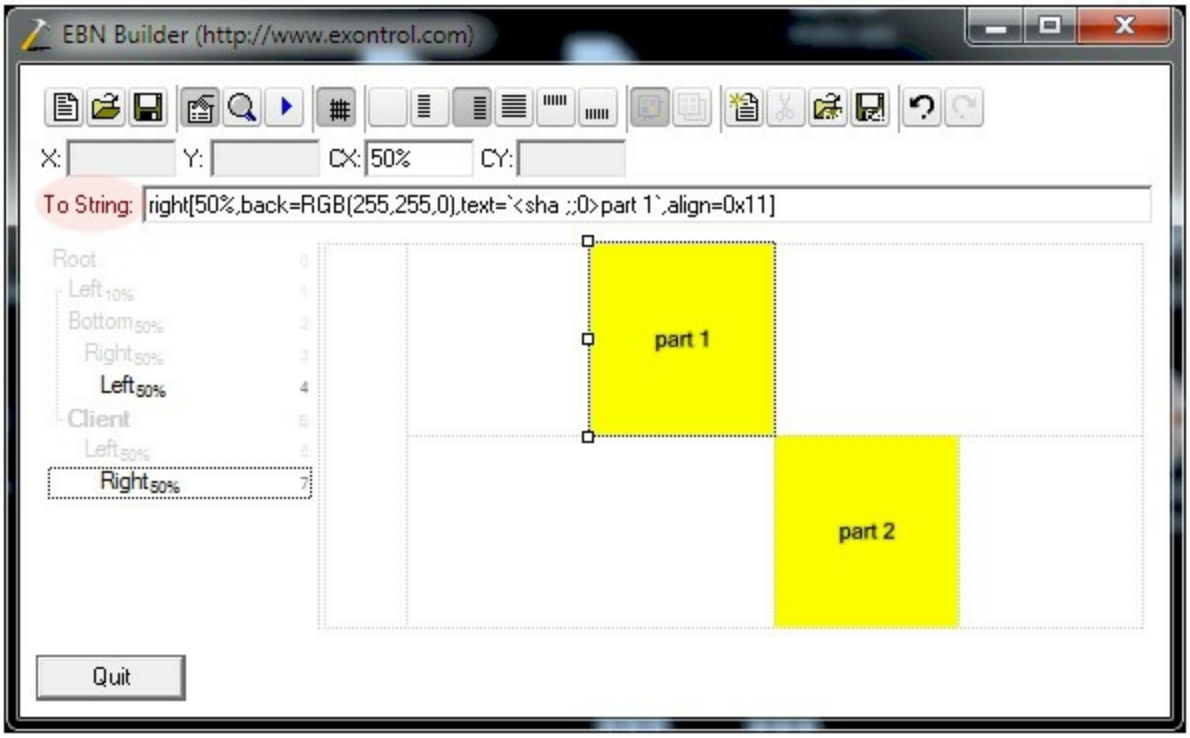
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



- "root[text=`caption <b>2`,align=0x22](client[text=`caption <b>1`,align=0x20])", shows the caption **1** aligned to the bottom-left side, and the caption **2** to the bottom-right side



*The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field as shown in the following screen shot:*



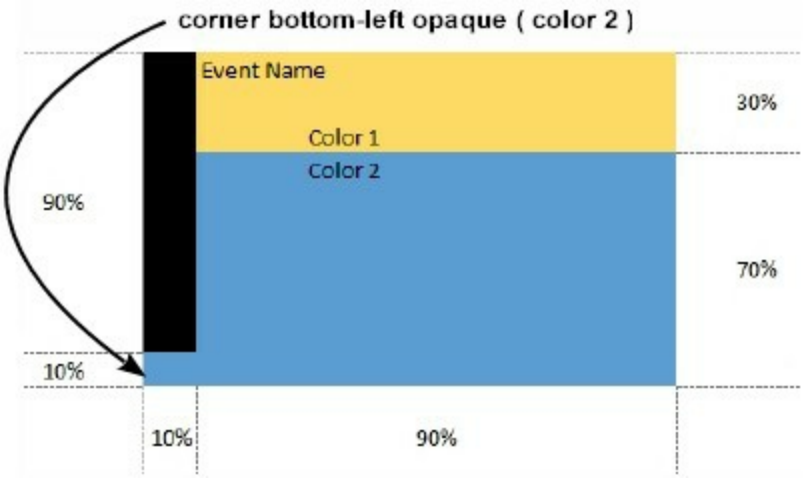
The **To String** field of the EBN Builder defines the **EBN String Format** that can be used on BackgroundExt property.

The **EBN String Format** syntax in BNF notation is defined like follows:

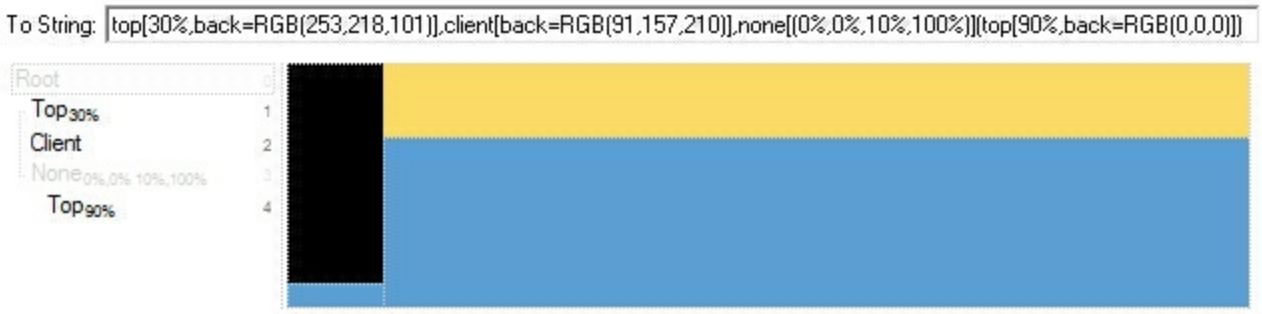
```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<client> ","] <attribute> [ "," <attributes> ] "]"
<client> ::= <expression> | <expression> "," <expression> "," <expression> ","
<expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> | <wordwrap> | <align> | <pattern> |
<patterncolor> | <frame> | <framethick> | <data> | <others>
<equal> ::= "="
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"  "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number> ")"
<string> ::= "`" <characters> "`" | "'" <characters> "'" | " <characters> "
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
<patterncolor> ::= "patterncolor" <equal> <color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

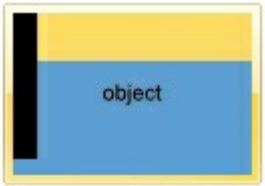
Now, lets say we have the following request to layout the colors on the objects:



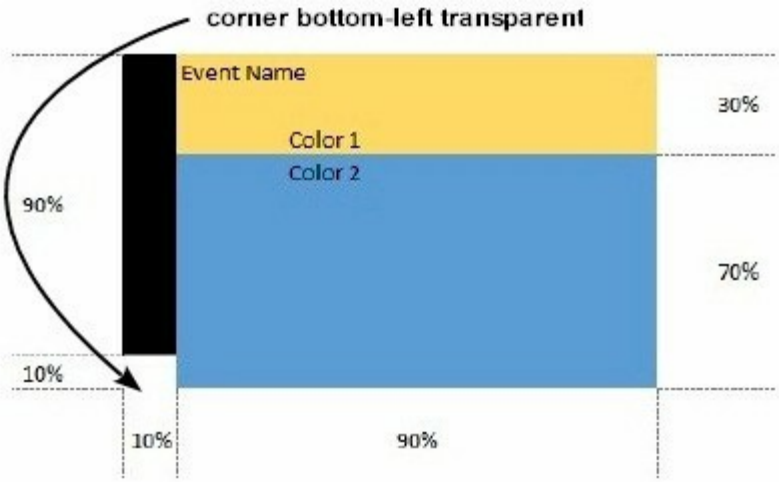
We define the BackgroundExt property such as "top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100
(top[90%,back=RGB(0,0,0)])", and it looks as:



so, if we apply to our object we got:

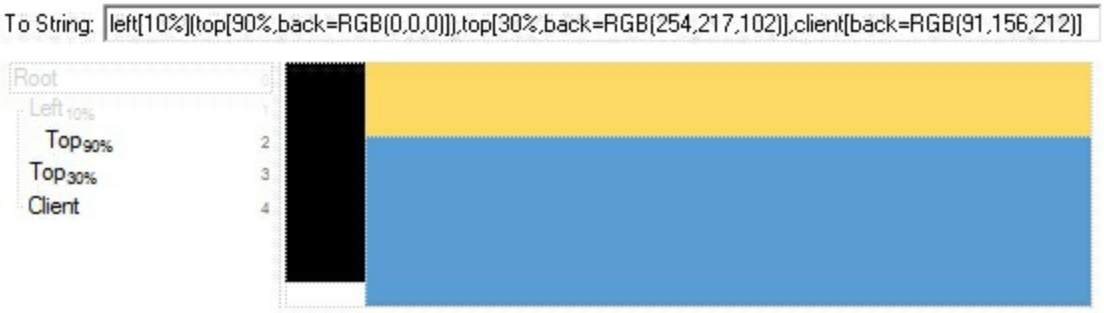


Now, lets say we have the following request to layout the colors on the objects:

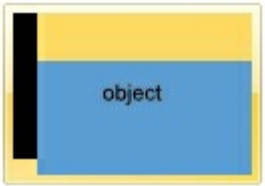


We define BackgroundExt property such as "left[10%]

(top[90%,back=RGB(0,0,0)]),top[30%,back=RGB(254,217,102)],client[back=RGB(91,156,2
and it looks as:

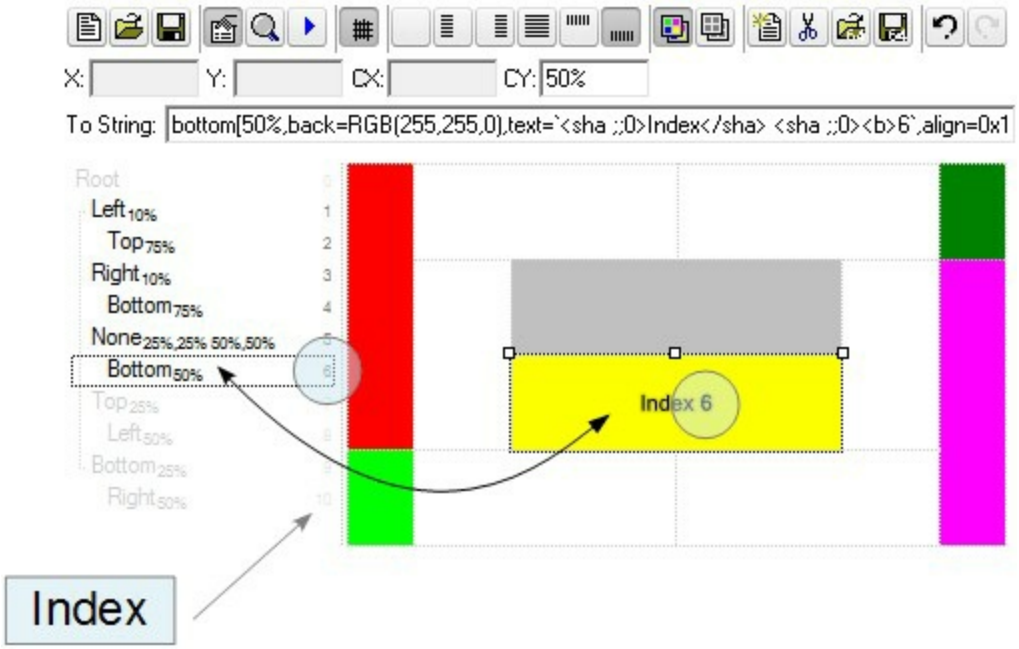


so, if we apply to our object we got:



property Frame.BackgroundExtValue(Index as IndexExtEnum, Property as BackgroundExtPropertyEnum) as Variant

Specifies at runtime, the value of the giving property for specified part of the background extension.

Type	Description
Index as IndexExtEnum	A Long expression that defines the index of the part that composes the EBN to be accessed / changed. The following screen shot shows where you can find Index of the parts:  The screen shot shows that the EBN contains 11 elements, so in this case the Index starts at 0 (root element) and ends on 10.
Property as BackgroundExtPropertyEnum	A BackgroundExtPropertyEnum expression that specifies the property to be changed as explained bellow.
Variant	A Variant expression that defines the part's value. The Type of the expression depending on the Property parameter as explained bellow.

Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExtValue property has no effect if the BackgroundExt property is empty (by default). *The idea is as follows: first you need to decide the layout of the UI to put on the object's background, using the*

BodyBackgroundExt property, and next (if required), you can change any property of any part of the background extension to a new value. In other words, let's say you have the same layout to be applied to some of your objects, so you specify the BodyBackgroundExt to be the same for them, and next use the BackgroundExtValue property to change particular properties (like back-color, size, position, anchor) for different objects.

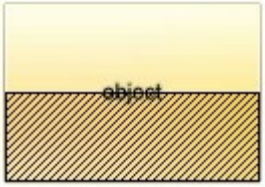
You can access/define/change the following UI properties of the element:

- **exBackColorExt**(1), Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000. (Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*
- **exClientExt**(2), Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect. Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent. (String/Numeric expression)*
- **exAnchorExt**(3), Specifies the object's alignment relative to its parent. *(Numeric expression)*
- **exTextExt**(4), Specifies the HTML text to be displayed on the object. *(String expression)*
- **exTextExtWordWrap**(5), Specifies that the object is wrapping the text. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *(Boolean expression)*
- **exTextExtAlignment**(6), Indicates the alignment of the text on the object. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag *(Numeric expression)*
- **exPatternExt**(7), Indicates the pattern to be shown on the object. The exPatternColorExt specifies the color to show the pattern. *(Numeric expression)*
- **exPatternColorExt**(8), Indicates the color to show the pattern on the object. The exPatternColorExt property has effect only if the exPatternExt property is not 0 (empty). The exFrameColorExt specifies the color to show the frame (the exPatternExt property includes the exFrame or exFrameThick flag). *(Color expression)*
- **exFrameColorExt**(9), Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. *(Color expression)*
- **exFrameThickExt**(11), Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property. *(Boolean expression)*
- **exUserDataExt**(12), Specifies an extra-data associated with the object. *(Variant*

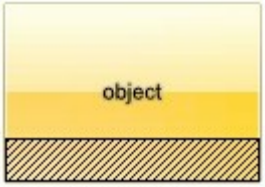
expression)

For instance, having the BodyBackgroundExt on "bottom[50%,pattern=6,frame]"

we got:



so let's change the percent of 50% to 25% like BackgroundExtValue(1,2) on "25%", where 1 indicates the first element after root, and 2 indicates the **exClientExt** property, we get:



In VB you should have the following syntax:

```
.BodyBackgroundExt = "bottom[50%,pattern=6,frame]"
.BackgroundExtValue(exIndexExt1, exClientExt) = "25%"
```

property Frame.Index as Long

Indicates the index of the current Frame object within the Frames collection.

Type	Description
Long	A Long expression that specifies the index of the frame.

The Index property indicates the index of the current Frame object within the Frames collection. The Count property returns the number of Frame objects in the collection. The Clear method removes all Frame objects. The Remove method removes a specified frame. The Visible property shows or hides a specified frame. You can use the Item and Count properties to enumerate the frames in the control, as well as **for each** statement.

property Frame.Nodes as String

Specifies the list of keys for the nodes to show the frame, separated by comma character. The node's key may ends with (all) to include all child, assistant nodes, with (child) to include the direct children of the node only, with (assistant) to include the assistant nodes of the node, with (group) to include all nodes in the same group, or it refers to the node itself.

Type	Description
String	A String expression that defines a collection of nodes that defines the frame. The list contains key of the nodes, separated by comma character as explained bellow. For instance "AK1,AK2" or "AK1(child)"

The Nodes property (equivalent with the Nodes parameter of the Add method) defines the list of nodes to be included in the frame, separated by comma character (,) as follows:

- **key**, indicates the node itself, not including the child, group or assistant nodes. The Key property of the Node defines the node's key.
- **key(all),** indicates all recursively child, group and assistant nodes of the node with the giving key.
- **key(child),** indicates all child nodes of the node with the giving key, not including any sub-child.
- **key(assistant),** indicates all assistant nodes of the node with the giving key. The AddAssistant property adds an assistant node.
- **key(group),** indicates all group nodes of the node with the giving key. The AddGroup property adds a new node to the same group.

For instance:

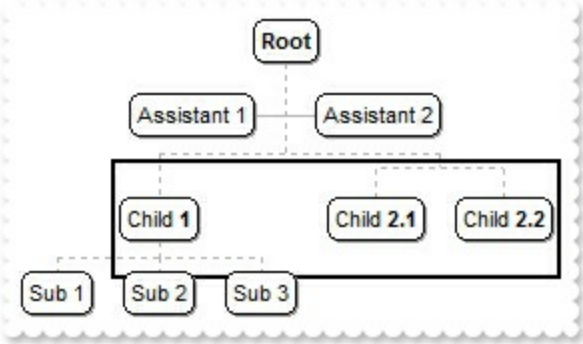
- "root(all)" defines all child, assistant, group of the root node, not including the root node itself.
- "root,root(all)" defines all child, assistant, group of the root node, including the root node too.

Once a new frame is added you can:

- specify whether the frame is shown on the back or on the front using the ShowOnBackground property
- define the padding of the frame using the Padding property
- define a solid or EBN background color to be displayed on the frame's background, using the BackColor property of the Frame object
- The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's

background.
- define a different border or pattern to be shown, using the Pattern property

The following screen shot shows a thick black-border around child nodes of the root node ("root(child)"):



property Frame.Padding(Edge as PaddingEdgeEnum) as Long

Returns or sets a value that indicates the padding of the frame.

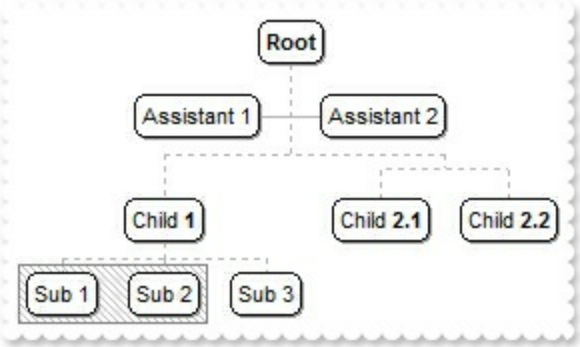
Type	Description
Edge as PaddingEdgeEnum	A PaddingEdgeEnum expression that specifies the edge to be updated / requested
Long	A long expression that defines the frame's padding

By default, the Padding(exPaddingAll) property is 4. Use the Padding property of the Frame to define the padding for specified frame. The Visible property specifies whether the frame is visible or shown. The ShowOnBackground property indicates whether the frame is displayed on the back or front of the chart.

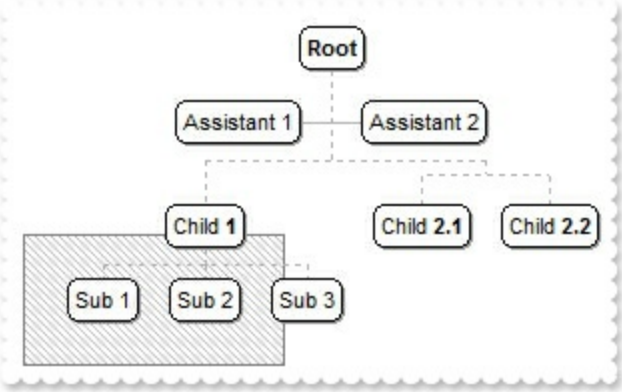
Once a new frame is added you can:

- specify whether the frame is shown on the back or on the front using the ShowOnBackground property
- define a solid or EBN background color to be displayed on the frame's background, using the BackColor property of the Frame object
- The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background.
- define a different border or pattern to be shown, using the Pattern property

The following screen shot shows the frame with default padding (4):



The following screen shot shows the frame with changed padding (22):


Root
Assistant 1
Assistant 2
Child 1
Child 2.1
Child 2.2
Sub 1
Sub 2
Sub 3

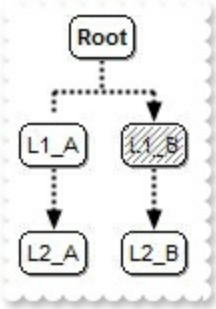
property Frame.Pattern as Pattern

Specifies the pattern of the frame.

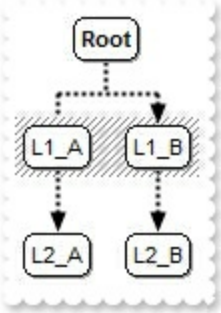
Type	Description
Pattern	A Pattern object that defines the pattern/borders to be shown on the frame.

By default, the frame shows a thick-black border around the nodes, so the Pattern.Type property is exPatternFrameThick. The ShowOnBackground property specifies whether the frame is shown on the back or front of the chart. The Visible property shows or hides a specified frame.

The following screen shot shows a pattern on the node:



The following screen shot shows a pattern on the frame:



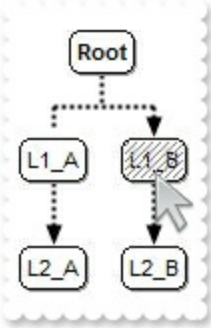
property Frame.ShowOnBackground as Boolean

Shows the frame on the control's background.

Type	Description
Boolean	A Boolean expression that specifies whether the frame is shown on the back or front of the chart.

By default, the ShowOnBackground property is True, which indicates that the frame is shown on the back of the chart. The ShowOnBackground property specifies whether the frame is shown on the back or front of the chart. The Visible property shows or hides a specified frame. The Remove method removes a specified frame. The Clear method removes all Frame objects.

The following sample shows how you can can draw a pattern on a node:



VBA (MS Access, Excell...)

```
With ChartView1
   .IndentSiblingY = 30
   .ShowLinksDir = True
   .PenWidthLink = 2
   .LinkColor = RGB(0,0,0)
   .AntiAliasing = True
   With .Nodes
      .Add("L1_A",,"LA").ShowLinkDir = False
      .Add "L1_B",,"LB"
      .Add "L2_A","LA","LA2"
      .Add "L2_B","LB","LB2"
   End With
   With .Frames.Add("LB")
      .Padding(-1) = -2
      With .Pattern
```

```
        .Type = 6
        .Color = RGB(128,128,128)
      End With
      .ShowOnBackground = False
    End With
End With
```

VB6

```
With ChartView1
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
      .Add("L1_A",,"LA").ShowLinkDir = False
      .Add "L1_B",,"LB"
      .Add "L2_A","LA","LA2"
      .Add "L2_B","LB","LB2"
    End With
    With .Frames.Add("LB")
      .Padding(exPaddingAll) = -2
      With .Pattern
        .Type = exPatternBDiagonal
        .Color = RGB(128,128,128)
      End With
      .ShowOnBackground = False
    End With
End With
```

VB.NET

```
With Exchartview1
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = Color.FromArgb(0,0,0)
```

```
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",,"LA").ShowLinkDir = False
        .Add("L1_B",,"LB")
        .Add("L2_A","LA","LA2")
        .Add("L2_B","LB","LB2")
    End With
    With .Frames.Add("LB")
        .set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,-2)
        With .Pattern
            .Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternBDiagonal
            .Color = Color.FromArgb(128,128,128)
        End With
        .ShowOnBackground = False
    End With
End With
```

VB.NET for /COM

```
With AxChartView1
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",,"LA").ShowLinkDir = False
        .Add("L1_B",,"LB")
        .Add("L2_A","LA","LA2")
        .Add("L2_B","LB","LB2")
    End With
    With .Frames.Add("LB")
        .Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll) = -2
        With .Pattern
            .Type = EXORGCHARTLib.PatternEnum.exPatternBDiagonal
            .Color = RGB(128,128,128)
        End With
```

```
    .ShowOnBackground = False
  End With
End With
```

C++

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

    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->PutIndentSiblingY(30);
spChartView1->PutShowLinksDir(VARIANT_TRUE);
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(RGB(0,0,0));
spChartView1->PutAntiAliasing(VARIANT_TRUE);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"L1_A",vtMissing,"LA",vtMissing,vtMissing)-
>PutShowLinkDir(VARIANT_FALSE);
   var_Nodes->Add(L"L1_B",vtMissing,"LB",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_A","LA","LA2",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_B","LB","LB2",vtMissing,vtMissing);
EXORGCHARTLib::IFramePtr var_Frame = spChartView1->GetFrames()->Add("LB");
   var_Frame->PutPadding(EXORGCHARTLib::exPaddingAll,-2);
   EXORGCHARTLib::IPatternPtr var_Pattern = var_Frame->GetPattern();
      var_Pattern->PutType(EXORGCHARTLib::exPatternBDiagonal);
      var_Pattern->PutColor(RGB(128,128,128));
   var_Frame->PutShowOnBackground(VARIANT_FALSE);
```

C++ Builder

```
ChartView1->IndentSiblingY = 30;
```

```
ChartView1->ShowLinksDir = true;
ChartView1->PenWidthLink = 2;
ChartView1->LinkColor = RGB(0,0,0);
ChartView1->AntiAliasing = true;
Exorgchartlib_tlb::INodesPtr var_Nodes = ChartView1->Nodes;
   var_Nodes->Add(L"L1_A",TNoParam(),TVariant("LA"),TNoParam(),TNoParam())-
>ShowLinkDir = false;
   var_Nodes->Add(L"L1_B",TNoParam(),TVariant("LB"),TNoParam(),TNoParam());
   var_Nodes->Add(L"L2_A",TVariant("LA"),TVariant("LA2"),TNoParam(),TNoParam());
   var_Nodes->Add(L"L2_B",TVariant("LB"),TVariant("LB2"),TNoParam(),TNoParam());
Exorgchartlib_tlb::IFramePtr var_Frame = ChartView1->Frames->Add(TVariant("LB"));
   var_Frame->set_Padding(Exorgchartlib_tlb::PaddingEdgeEnum::exPaddingAll,-2);
   Exorgchartlib_tlb::IPatternPtr var_Pattern = var_Frame->Pattern;
      var_Pattern->Type = Exorgchartlib_tlb::PatternEnum::exPatternBDiagonal;
      var_Pattern->Color = RGB(128,128,128);
   var_Frame->ShowOnBackground = false;
```

C#

```
exchartview1.IndentSiblingY = 30;
exchartview1.ShowLinksDir = true;
exchartview1.PenWidthLink = 2;
exchartview1.LinkColor = Color.FromArgb(0,0,0);
exchartview1.AntiAliasing = true;
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
   var_Nodes.Add("L1_A",null,"LA",null,null).ShowLinkDir = false;
   var_Nodes.Add("L1_B",null,"LB",null,null);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
   var_Nodes.Add("L2_B","LB","LB2",null,null);
exontrol.EXORGCHARTLib.Frame var_Frame = exchartview1.Frames.Add("LB");

var_Frame.set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,-2);

   exontrol.EXORGCHARTLib.Pattern var_Pattern = var_Frame.Pattern;
      var_Pattern.Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternBDiagonal;
      var_Pattern.Color = Color.FromArgb(128,128,128);
```

```
var_Frame.ShowOnBackground = false;
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   ChartView1.IndentSiblingY = 30;
   ChartView1.ShowLinksDir = true;
   ChartView1.PenWidthLink = 2;
   ChartView1.LinkColor = 0;
   ChartView1.AntiAliasing = true;
   var var_Nodes = ChartView1.Nodes;
      var_Nodes.Add("L1_A",null,"LA",null,null).ShowLinkDir = false;
      var_Nodes.Add("L1_B",null,"LB",null,null);
      var_Nodes.Add("L2_A","LA","LA2",null,null);
      var_Nodes.Add("L2_B","LB","LB2",null,null);
   var var_Frame = ChartView1.Frames.Add("LB");
      var_Frame.Padding(-1) = -2;
      var var_Pattern = var_Frame.Pattern;
         var_Pattern.Type = 6;
         var_Pattern.Color = 8421504;
      var_Frame.ShowOnBackground = false;
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>
```

```
<SCRIPT LANGUAGE="VBScript">
Function Init()
   With ChartView1
      .IndentSiblingY = 30
      .ShowLinksDir = True
      .PenWidthLink = 2
      .LinkColor = RGB(0,0,0)
      .AntiAliasing = True
      With .Nodes
         .Add("L1_A",,"LA").ShowLinkDir = False
         .Add "L1_B",,"LB"
         .Add "L2_A","LA","LA2"
         .Add "L2_B","LB","LB2"
      End With
      With .Frames.Add("LB")
         .Padding(-1) = -2
         With .Pattern
            .Type = 6
            .Color = RGB(128,128,128)
         End With
         .ShowOnBackground = False
      End With
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axChartView1.IndentSiblingY = 30;
axChartView1.ShowLinksDir = true;
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
axChartView1.AntiAliasing = true;
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
```

```
   var_Nodes.Add("L1_A",null,"LA",null,null).ShowLinkDir = false;
   var_Nodes.Add("L1_B",null,"LB",null,null);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
   var_Nodes.Add("L2_B","LB","LB2",null,null);
EXORGCHARTLib.Frame var_Frame = axChartView1.Frames.Add("LB");
   var_Frame.set_Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,-2);
   EXORGCHARTLib.Pattern var_Pattern = var_Frame.Pattern;
      var_Pattern.Type = EXORGCHARTLib.PatternEnum.exPatternBDiagonal;
      var_Pattern.Color =
(uint)ColorTranslator.ToWin32(Color.FromArgb(128,128,128));
   var_Frame.ShowOnBackground = false;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Frame,com_Node,com_Nodes,com_Pattern;
   anytype var_Frame,var_Node,var_Nodes,var_Pattern;
   ;

   super();

   exchartview1.IndentSiblingY(30);
   exchartview1.ShowLinksDir(true);
   exchartview1.PenWidthLink(2);
   exchartview1.LinkColor(WinApi::RGB2int(0,0,0));
   exchartview1.AntiAliasing(true);
   var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
      var_Node = COM::createFromObject(com_Nodes.Add("L1_A",,"LA")); com_Node
= var_Node;
      com_Node.ShowLinkDir(false);
      com_Nodes.Add("L1_B",,"LB");
      com_Nodes.Add("L2_A","LA","LA2");
      com_Nodes.Add("L2_B","LB","LB2");
   var_Frame = COM::createFromObject(exchartview1.Frames()).Add("LB");
com_Frame = var_Frame;
```

```
    com_Frame.Padding(-1/*exPaddingAll*/,-2);
    var_Pattern = com_Frame.Pattern(); com_Pattern = var_Pattern;
      com_Pattern.Type(6/*exPatternBDiagonal*/);
      com_Pattern.Color(WinApi::RGB2int(128,128,128));
    com_Frame.ShowOnBackground(false);
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
  IndentSiblingY := 30;
  ShowLinksDir := True;
  PenWidthLink := 2;
  LinkColor := Color.FromArgb(0,0,0);
  AntiAliasing := True;
  with Nodes do
  begin
    Add('L1_A',Nil,'LA',Nil,Nil).ShowLinkDir := False;
    Add('L1_B',Nil,'LB',Nil,Nil);
    Add('L2_A','LA','LA2',Nil,Nil);
    Add('L2_B','LB','LB2',Nil,Nil);
  end;
  with Frames.Add('LB') do
  begin
    Padding[EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll] := -2;
    with Pattern do
    begin
      Type := EXORGCHARTLib.PatternEnum.exPatternBDiagonal;
      Color := $808080;
    end;
    ShowOnBackground := False;
  end;
end
```

Delphi (standard)

```
with ChartView1 do
```

```
begin
  IndentSiblingY := 30;
  ShowLinksDir := True;
  PenWidthLink := 2;
  LinkColor := RGB(0,0,0);
  AntiAliasing := True;
  with Nodes do
  begin
    Add('L1_A',Null,'LA',Null,Null).ShowLinkDir := False;
    Add('L1_B',Null,'LB',Null,Null);
    Add('L2_A','LA','LA2',Null,Null);
    Add('L2_B','LB','LB2',Null,Null);
  end;
  with Frames.Add('LB') do
  begin
    Padding[EXORGCHARTLib_TLB.exPaddingAll] := -2;
    with Pattern do
    begin
      Type := EXORGCHARTLib_TLB.exPatternBDiagonal;
      Color := $808080;
    end;
    ShowOnBackground := False;
  end;
end
```

VFP

```
with thisform.ChartView1
  .IndentSiblingY = 30
  .ShowLinksDir = .T.
  .PenWidthLink = 2
  .LinkColor = RGB(0,0,0)
  .AntiAliasing = .T.
  with .Nodes
    .Add("L1_A",Null,"LA").ShowLinkDir = .F.
    .Add("L1_B",Null,"LB")
    .Add("L2_A","LA","LA2")
```

```
      .Add("L2_B","LB","LB2")
   endwith
   with .Frames.Add("LB")
      .Padding(-1) = -2
      with .Pattern
         .Type = 6
         .Color = RGB(128,128,128)
      endwith
      .ShowOnBackground = .F.
   endwith
endwith
```

dBASE Plus

```
local oChartView,var_Frame,var_Node,var_Nodes,var_Pattern

oChartView = form.EXORGCHARTACTIVEXCONTROL1.nativeObject
oChartView.IndentSiblingY = 30
oChartView.ShowLinksDir = true
oChartView.PenWidthLink = 2
oChartView.LinkColor = 0x0
oChartView.AntiAliasing = true
var_Nodes = oChartView.Nodes
   // var_Nodes.Add("L1_A",null,"LA").ShowLinkDir = false
   var_Node = var_Nodes.Add("L1_A",null,"LA")
   with (oChartView)
      TemplateDef = [dim var_Node]
      TemplateDef = var_Node
      Template = [var_Node.ShowLinkDir = False]
   endwith
   var_Nodes.Add("L1_B",null,"LB")
   var_Nodes.Add("L2_A","LA","LA2")
   var_Nodes.Add("L2_B","LB","LB2")
var_Frame = oChartView.Frames.Add("LB")
   // var_Frame.Padding(-1) = -2
   with (oChartView)
      TemplateDef = [dim var_Frame]
```

```
    TemplateDef = var_Frame
    Template = [var_Frame.Padding(-1) = -2]
  endwith
  var_Pattern = var_Frame.Pattern
    var_Pattern.Type = 6
    var_Pattern.Color = 0x808080
  var_Frame.ShowOnBackground = false
```

XBasic (Alpha Five)

```
Dim oChartView as P
Dim var_Frame as P
Dim var_Node as local
Dim var_Nodes as P
Dim var_Pattern as P

oChartView = topparent:CONTROL_ACTIVEX1.activex
oChartView.IndentSiblingY = 30
oChartView.ShowLinksDir = .t.
oChartView.PenWidthLink = 2
oChartView.LinkColor = 0
oChartView.AntiAliasing = .t.
var_Nodes = oChartView.Nodes
  ' var_Nodes.Add("L1_A",,"LA").ShowLinkDir = .f.
  var_Node = var_Nodes.Add("L1_A",,"LA")
  oChartView.TemplateDef = "dim var_Node"
  oChartView.TemplateDef = var_Node
  oChartView.Template = "var_Node.ShowLinkDir = False"

  var_Nodes.Add("L1_B",,"LB")
  var_Nodes.Add("L2_A","LA","LA2")
  var_Nodes.Add("L2_B","LB","LB2")
var_Frame = oChartView.Frames.Add("LB")
  ' var_Frame.Padding(-1) = -2
  oChartView.TemplateDef = "dim var_Frame"
  oChartView.TemplateDef = var_Frame
```

```
oChartView.Template = "var_Frame.Padding(-1) = -2"

var_Pattern = var_Frame.Pattern
   var_Pattern.Type = 6
   var_Pattern.Color = 8421504
var_Frame.ShowOnBackground = .f.
```

Visual Objects

```
local var_Frame as IFrame
local var_Nodes as INodes
local var_Pattern as IPattern

oDCOCX_Exontrol1:IndentSiblingY := 30
oDCOCX_Exontrol1:ShowLinksDir := true
oDCOCX_Exontrol1:PenWidthLink := 2
oDCOCX_Exontrol1:LinkColor := RGB(0,0,0)
oDCOCX_Exontrol1:AntiAliasing := true
var_Nodes := oDCOCX_Exontrol1:Nodes
   var_Nodes:Add("L1_A",nil,"LA",nil,nil):ShowLinkDir := false
   var_Nodes:Add("L1_B",nil,"LB",nil,nil)
   var_Nodes:Add("L2_A","LA","LA2",nil,nil)
   var_Nodes:Add("L2_B","LB","LB2",nil,nil)
var_Frame := oDCOCX_Exontrol1:Frames:Add("LB")
   var_Frame:[Padding,exPaddingAll] := -2
   var_Pattern := var_Frame:Pattern
      var_Pattern:Type := exPatternBDiagonal
      var_Pattern:Color := RGB(128,128,128)
   var_Frame:ShowOnBackground := false
```

PowerBuilder

```
OleObject oChartView,var_Frame,var_Nodes,var_Pattern

oChartView = ole_1.Object
oChartView.IndentSiblingY = 30
```

```
oChartView.ShowLinksDir = true
oChartView.PenWidthLink = 2
oChartView.LinkColor = RGB(0,0,0)
oChartView.AntiAliasing = true
var_Nodes = oChartView.Nodes
   var_Nodes.Add("L1_A",,"LA").ShowLinkDir = false
   var_Nodes.Add("L1_B",,"LB")
   var_Nodes.Add("L2_A","LA","LA2")
   var_Nodes.Add("L2_B","LB","LB2")
var_Frame = oChartView.Frames.Add("LB")
   var_Frame.Padding(-1,-2)
   var_Pattern = var_Frame.Pattern
      var_Pattern.Type = 6
      var_Pattern.Color = RGB(128,128,128)
   var_Frame.ShowOnBackground = false
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Set ComIndentSiblingY to 30
   Set ComShowLinksDir to True
   Set ComPenWidthLink to 2
   Set ComLinkColor to (RGB(0,0,0))
   Set ComAntiAliasing to True
   Variant voNodes
   Get ComNodes to voNodes
   Handle hoNodes
   Get Create (RefClass(cComNodes)) to hoNodes
   Set pvComObject of hoNodes to voNodes
      Variant voNode
      Get ComAdd of hoNodes "L1_A" "LA" Nothing Nothing to voNode
      Handle hoNode
      Get Create (RefClass(cComNode)) to hoNode
      Set pvComObject of hoNode to voNode
         Set ComShowLinkDir of hoNode to False
```

```
        Send Destroy to hoNode
        Get ComAdd of hoNodes "L1_B" "LB" Nothing Nothing to Nothing
        Get ComAdd of hoNodes "L2_A" "LA" "LA2" Nothing Nothing to Nothing
        Get ComAdd of hoNodes "L2_B" "LB" "LB2" Nothing Nothing to Nothing
    Send Destroy to hoNodes
    Variant voFrames
    Get ComFrames to voFrames
    Handle hoFrames
    Get Create (RefClass(cComFrames)) to hoFrames
    Set pvComObject of hoFrames to voFrames
        Variant voFrame
        Get ComAdd of hoFrames "LB" to voFrame
        Handle hoFrame
        Get Create (RefClass(cComFrame)) to hoFrame
        Set pvComObject of hoFrame to voFrame
            Set ComPadding of hoFrame OLEexPaddingAll to -2
            Variant voPattern
            Get ComPattern of hoFrame to voPattern
            Handle hoPattern
            Get Create (RefClass(cComPattern)) to hoPattern
            Set pvComObject of hoPattern to voPattern
                Set ComType of hoPattern to OLEexPatternBDiagonal
                Set ComColor of hoPattern to (RGB(128,128,128))
            Send Destroy to hoPattern
            Set ComShowOnBackground of hoFrame to False
        Send Destroy to hoFrame
    Send Destroy to hoFrames
End_Procedure
```

XBase++

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

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

```
   LOCAL oChartView
   LOCAL oFrame
   LOCAL oNodes
   LOCAL oPattern

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

   oChartView := XbpActiveXControl():new( oForm:drawingArea )
   oChartView:CLSID  := "Exontrol.ChartView.1" /*{F4DFE455-01FE-420E-A088-
64346DCC3791}*/
   oChartView:create(,, {10,60},{610,370} )

      oChartView:IndentSiblingY := 30
      oChartView:ShowLinksDir := .T.
      oChartView:PenWidthLink := 2
      oChartView:SetProperty("LinkColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oChartView:AntiAliasing := .T.
      oNodes := oChartView:Nodes()
         oNodes:Add("L1_A",,"LA"):ShowLinkDir := .F.
         oNodes:Add("L1_B",,"LB")
         oNodes:Add("L2_A","LA","LA2")
         oNodes:Add("L2_B","LB","LB2")
      oFrame := oChartView:Frames():Add("LB")
         oFrame:SetProperty("Padding",-1/*exPaddingAll*/,-2)
         oPattern := oFrame:Pattern()
            oPattern:Type := 6/*exPatternBDiagonal*/
            oPattern:SetProperty("Color",AutomationTranslateColor( GraMakeRGBColor
( { 128,128,128 } )  , .F. ))
         oFrame:ShowOnBackground := .F.

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

```
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO
RETURN
```

property Frame.Visible as Boolean

Shows or hides the frame.

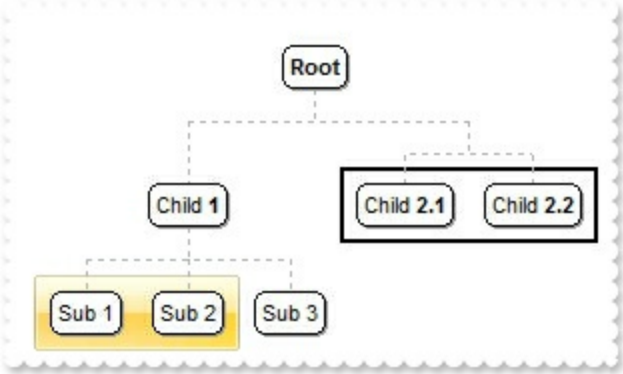
Type	Description
Boolean	A Boolean expression that specifies whether the frame is shown or hidden.

By default, the Visible property is True, which indicates that the frame is shown / visible by default. The Visible property shows or hides a specified frame. The ShowOnBackground property specifies whether the frame is shown on the back or front of the chart. The Remove method removes a specified frame. The Clear method removes all Frame objects.

Frames object

The Frames object holds a collection of Frame objects. The Frames property of the control gives access to the Frames collection.

The following screen shot shows different frames on nodes:



A Frame is defined by an union of nodes, and can:

- specify whether the frame is shown on the back or on the front using the ShowOnBackground property
- define the padding of the frame using the Padding property
- define a solid or EBN background color to be displayed on the frame's background, using the BackColor property of the Frame object
- The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background.
- define a different border or pattern to be shown, using the Pattern property

The Frames collection supports the following properties and methods:

Name	Description
Add	Adds an Frame 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 elements in the collection.
Item	Returns a specific Frame of the Frames collection, giving its index.
Remove	Removes a specific member from the Frames collection, giving its index or reference.

method Frames.Add ([Nodes as Variant])

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

Type	Description
Nodes as Variant	A String expression that defines a collection of nodes that defines the frame. The list contains key of the nodes, separated by comma character as explained bellow. For instance "AK1,AK2" or "AK1(child)"

Return	Description
Frame	A Frame object that holds information about frame's drawing options.

The Add method adds a new frame to the chart. A frame is defined by a union of nodes. The Nodes property gives access to the chart's nodes collection. By default, the frame shows a thick black-border around giving nodes. The Visible property specifies whether the frame is visible or shown. The ShowOnBackground property indicates whether the frame is displayed on the back or front of the chart.

The Nodes parameter (equivalent with the Nodes property of the Frame object) of the Add method defines the list of nodes to be included in the frame, separated by comma character (,) as follows:

- **key**, indicates the node itself, not including the child, group or assistant nodes. The Key property of the Node defines the node's key.
- **key(all),** indicates all recursively child, group and assistant nodes of the node with the giving key.
- **key(child),** indicates all child nodes of the node with the giving key, not including any sub-child.
- **key(assistant),** indicates all assistant nodes of the node with the giving key. The AddAssistant property adds an assistant node.
- **key(group),** indicates all group nodes of the node with the giving key. The AddGroup property adds a new node to the same group.

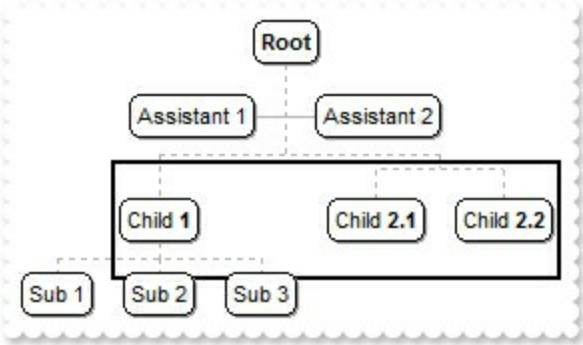
For instance:

- "root(all)" defines all child, assistant, group of the root node, not including the root node itself.
- "root,root(all)" defines all child, assistant, group of the root node, including the root node too.

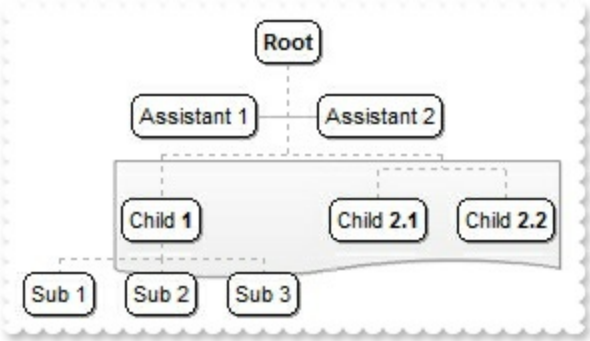
Once a new frame is added you can:

- specify whether the frame is shown on the back or on the front using the ShowOnBackground property
- define the padding of the frame using the Padding property
- define a solid or EBN background color to be displayed on the frame's background, using the BackColor property of the Frame object
- The BackgroundExt property of the Frame object, defines unlimited options to show any HTML text, images, colors, EBNs, patterns, borders anywhere on the frame's background.
- define a different border or pattern to be shown, using the Pattern property

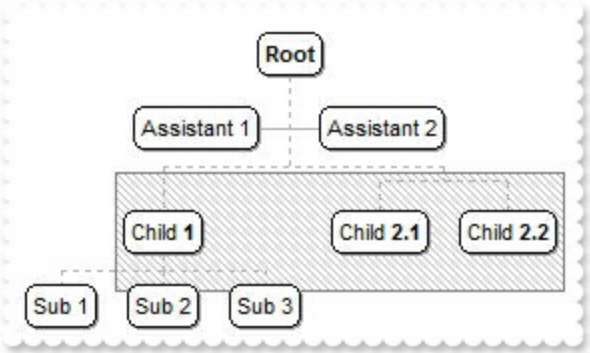
The following screen shot shows a thick black-border around child nodes of the root node ("root(child)"):



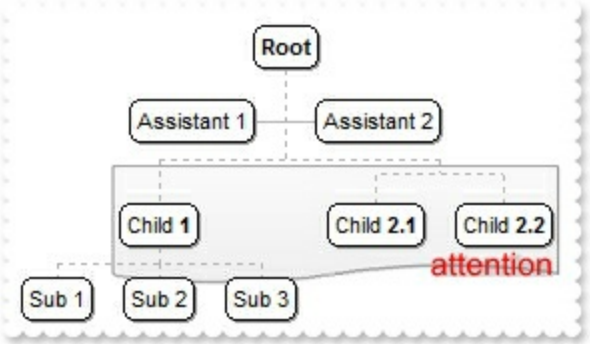
The following screen shot shows an EBN object around child nodes of the root node ("root(child)"):



The following screen shot shows a pattern + border around child nodes of the root node ("root(child)"):



The following screen shot shows a a text around child nodes of the root node ("root(child)"):



The following sample shows how you can visually group nodes that have the same parent but also have other siblings:

VBA (MS Access, Excell...)

```
With ChartView1
   .BeginUpdate
   With .Root
      .AddAssistant "Assistant 1"
      .AddAssistant "Assistant 2"
   End With
   With .Nodes
      .Add "Child <b>1</b>",,"1234"
      .Add "Sub 1","1234","AK1"
      .Add "Sub 2","1234","AK2"
      .Add "Sub 3","1234"
      .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
   End With
   With .Frames.Add("AK1,AK2").Pattern
      .Type = 261 ' PatternEnum.exPatternFrame Or PatternEnum.exPatternFDiagonal
```

```
        .Color = RGB(190,190,190)
        .FrameColor = RGB(128,128,128)
    End With
    .EndUpdate
End With
```

VB6

```
With ChartView1
    .BeginUpdate
    With .Root
        .AddAssistant "Assistant 1"
        .AddAssistant "Assistant 2"
    End With
    With .Nodes
        .Add "Child <b>1</b>",,"1234"
        .Add "Sub 1","1234","AK1"
        .Add "Sub 2","1234","AK2"
        .Add "Sub 3","1234"
        .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
    End With
    With .Frames.Add("AK1,AK2").Pattern
        .Type = PatternEnum.exPatternFrame Or PatternEnum.exPatternFDiagonal
        .Color = RGB(190,190,190)
        .FrameColor = RGB(128,128,128)
    End With
    .EndUpdate
End With
```

VB.NET

```
With Exchartview1
    .BeginUpdate()
    With .Root
        .AddAssistant("Assistant 1")
        .AddAssistant("Assistant 2")
    End With
    With .Nodes
```

```
        .Add("Child <b>1</b>",,"1234")
        .Add("Sub 1","1234","AK1")
        .Add("Sub 2","1234","AK2")
        .Add("Sub 3","1234")
        .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
    End With
    With .Frames.Add("AK1,AK2").Pattern
        .Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternFrame Or
exontrol.EXORGCHARTLib.PatternEnum.exPatternFDiagonal
        .Color = Color.FromArgb(190,190,190)
        .FrameColor = Color.FromArgb(128,128,128)
    End With
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxChartView1
    .BeginUpdate()
    With .Root
        .AddAssistant("Assistant 1")
        .AddAssistant("Assistant 2")
    End With
    With .Nodes
        .Add("Child <b>1</b>",,"1234")
        .Add("Sub 1","1234","AK1")
        .Add("Sub 2","1234","AK2")
        .Add("Sub 3","1234")
        .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
    End With
    With .Frames.Add("AK1,AK2").Pattern
        .Type = EXORGCHARTLib.PatternEnum.exPatternFrame Or
EXORGCHARTLib.PatternEnum.exPatternFDiagonal
        .Color = RGB(190,190,190)
        .FrameColor = RGB(128,128,128)
    End With
    .EndUpdate()
```

```
End With
```

C++

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

    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->BeginUpdate();
EXORGCHARTLib::INodePtr var_Node = spChartView1->GetRoot();
   var_Node->AddAssistant(L"Assistant 1",vtMissing,vtMissing);
   var_Node->AddAssistant(L"Assistant 2",vtMissing,vtMissing);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"Child <b>1</b>",vtMissing,"1234",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 1","1234","AK1",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 2","1234","AK2",vtMissing,vtMissing);
   var_Nodes->Add(L"Sub 3","1234",vtMissing,vtMissing,vtMissing);
   var_Nodes->Add(L"Child <b>2.1</b>",vtMissing,vtMissing,vtMissing,vtMissing)-
>AddGroup(L"Child <b>2.2</b>",vtMissing,vtMissing);
EXORGCHARTLib::IPatternPtr var_Pattern = spChartView1->GetFrames()-
>Add("AK1,AK2")->GetPattern();
   var_Pattern-
>PutType(EXORGCHARTLib::PatternEnum(EXORGCHARTLib::exPatternFrame |
EXORGCHARTLib::exPatternFDiagonal));
   var_Pattern->PutColor(RGB(190,190,190));
   var_Pattern->PutFrameColor(RGB(128,128,128));
spChartView1->EndUpdate();
```

C++ Builder

```
ChartView1->BeginUpdate();
```

```
Exorgchartlib_tlb::INodePtr var_Node = ChartView1->Root;
   var_Node->AddAssistant(L"Assistant 1",TNoParam(),TNoParam());
   var_Node->AddAssistant(L"Assistant 2",TNoParam(),TNoParam());
Exorgchartlib_tlb::INodesPtr var_Nodes = ChartView1->Nodes;
   var_Nodes->Add(L"Child
<b>1</b>",TNoParam(),TVariant("1234"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub
1",TVariant("1234"),TVariant("AK1"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub
2",TVariant("1234"),TVariant("AK2"),TNoParam(),TNoParam());
   var_Nodes->Add(L"Sub 3",TVariant("1234"),TNoParam(),TNoParam(),TNoParam());
   var_Nodes->Add(L"Child
<b>2.1</b>",TNoParam(),TNoParam(),TNoParam(),TNoParam())->AddGroup(L"Child
<b>2.2</b>",TNoParam(),TNoParam());
Exorgchartlib_tlb::IPatternPtr var_Pattern = ChartView1->Frames-
>Add(TVariant("AK1,AK2"))->Pattern;
   var_Pattern->Type = Exorgchartlib_tlb::PatternEnum::exPatternFrame |
Exorgchartlib_tlb::PatternEnum::exPatternFDiagonal;
   var_Pattern->Color = RGB(190,190,190);
   var_Pattern->FrameColor = RGB(128,128,128);
ChartView1->EndUpdate();
```

C#

```
exchartview1.BeginUpdate();
exontrol.EXORGCHARTLib.Node var_Node = exchartview1.Root;
   var_Node.AddAssistant("Assistant 1",null,null);
   var_Node.AddAssistant("Assistant 2",null,null);
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
   var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
   var_Nodes.Add("Sub 1","1234","AK1",null,null);
   var_Nodes.Add("Sub 2","1234","AK2",null,null);
   var_Nodes.Add("Sub 3","1234",null,null,null);
   var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
exontrol.EXORGCHARTLib.Pattern var_Pattern =
```

```
exchartview1.Frames.Add("AK1,AK2").Pattern;
   var_Pattern.Type = exontrol.EXORGCHARTLib.PatternEnum.exPatternFrame |
exontrol.EXORGCHARTLib.PatternEnum.exPatternFDiagonal;
   var_Pattern.Color = Color.FromArgb(190,190,190);
   var_Pattern.FrameColor = Color.FromArgb(128,128,128);
exchartview1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   ChartView1.BeginUpdate();
   var var_Node = ChartView1.Root;
      var_Node.AddAssistant("Assistant 1",null,null);
      var_Node.AddAssistant("Assistant 2",null,null);
   var var_Nodes = ChartView1.Nodes;
      var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
      var_Nodes.Add("Sub 1","1234","AK1",null,null);
      var_Nodes.Add("Sub 2","1234","AK2",null,null);
      var_Nodes.Add("Sub 3","1234",null,null,null);
      var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
   var var_Pattern = ChartView1.Frames.Add("AK1,AK2").Pattern;
      var_Pattern.Type = 261;
      var_Pattern.Color = 12500670;
      var_Pattern.FrameColor = 8421504;
   ChartView1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With ChartView1
      .BeginUpdate
      With .Root
         .AddAssistant "Assistant 1"
         .AddAssistant "Assistant 2"
      End With
      With .Nodes
         .Add "Child <b>1</b>",,"1234"
         .Add "Sub 1","1234","AK1"
         .Add "Sub 2","1234","AK2"
         .Add "Sub 3","1234"
         .Add("Child <b>2.1</b>").AddGroup "Child <b>2.2</b>"
      End With
      With .Frames.Add("AK1,AK2").Pattern
         .Type = 261 ' PatternEnum.exPatternFrame Or
PatternEnum.exPatternFDiagonal
         .Color = RGB(190,190,190)
         .FrameColor = RGB(128,128,128)
      End With
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axChartView1.BeginUpdate();
EXORGCHARTLib.Node var_Node = axChartView1.Root;
```

```
   var_Node.AddAssistant("Assistant 1",null,null);
   var_Node.AddAssistant("Assistant 2",null,null);
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   var_Nodes.Add("Child <b>1</b>",null,"1234",null,null);
   var_Nodes.Add("Sub 1","1234","AK1",null,null);
   var_Nodes.Add("Sub 2","1234","AK2",null,null);
   var_Nodes.Add("Sub 3","1234",null,null,null);
   var_Nodes.Add("Child <b>2.1</b>",null,null,null,null).AddGroup("Child
<b>2.2</b>",null,null);
EXORGCHARTLib.Pattern var_Pattern =
axChartView1.Frames.Add("AK1,AK2").Pattern;
   var_Pattern.Type = EXORGCHARTLib.PatternEnum.exPatternFrame |
EXORGCHARTLib.PatternEnum.exPatternFDiagonal;
   var_Pattern.Color = (uint)ColorTranslator.ToWin32(Color.FromArgb(190,190,190));
   var_Pattern.FrameColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(128,128,128));
axChartView1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Frame,com_Node,com_Node1,com_Nodes,com_Pattern;
   anytype var_Frame,var_Node,var_Node1,var_Nodes,var_Pattern;
   ;

   super();

   exchartview1.BeginUpdate();
   var_Node = exchartview1.Root(); com_Node = var_Node;
      com_Node.AddAssistant("Assistant 1");
      com_Node.AddAssistant("Assistant 2");
   var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
      com_Nodes.Add("Child <b>1</b>",,"1234");
      com_Nodes.Add("Sub 1","1234","AK1");
      com_Nodes.Add("Sub 2","1234","AK2");
```

```
      com_Nodes.Add("Sub 3","1234");
      var_Node1 = COM::createFromObject(com_Nodes.Add("Child <b>2.1</b>"));
com_Node1 = var_Node1;
      com_Node1.AddGroup("Child <b>2.2</b>");
   var_Frame = COM::createFromObject(exchartview1.Frames()).Add("AK1,AK2");
com_Frame = var_Frame;
   var_Pattern = com_Frame.Pattern(); com_Pattern = var_Pattern;
      com_Pattern.Type(261/*exPatternFrame | exPatternFDiagonal*/);
      com_Pattern.Color(WinApi::RGB2int(190,190,190));
      com_Pattern.FrameColor(WinApi::RGB2int(128,128,128));
   exchartview1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
   BeginUpdate();
   with Root do
   begin
      AddAssistant('Assistant 1',Nil,Nil);
      AddAssistant('Assistant 2',Nil,Nil);
   end;
   with Nodes do
   begin
      Add('Child <b>1</b>',Nil,'1234',Nil,Nil);
      Add('Sub 1','1234','AK1',Nil,Nil);
      Add('Sub 2','1234','AK2',Nil,Nil);
      Add('Sub 3','1234',Nil,Nil,Nil);
      Add('Child <b>2.1</b>',Nil,Nil,Nil,Nil).AddGroup('Child <b>2.2</b>',Nil,Nil);
   end;
   with Frames.Add('AK1,AK2').Pattern do
   begin
      Type := Integer(EXORGCHARTLib.PatternEnum.exPatternFrame) Or
Integer(EXORGCHARTLib.PatternEnum.exPatternFDiagonal);
      Color := $bebebe;
      FrameColor := $808080;
```

```
    end;
    EndUpdate();
end
```

Delphi (standard)

```
with ChartView1 do
begin
    BeginUpdate();
    with Root do
    begin
        AddAssistant('Assistant 1',Null,Null);
        AddAssistant('Assistant 2',Null,Null);
    end;
    with Nodes do
    begin
        Add('Child <b>1</b>',Null,'1234',Null,Null);
        Add('Sub 1','1234','AK1',Null,Null);
        Add('Sub 2','1234','AK2',Null,Null);
        Add('Sub 3','1234',Null,Null,Null);
        Add('Child <b>2.1</b>',Null,Null,Null,Null).AddGroup('Child
<b>2.2</b>',Null,Null);
    end;
    with Frames.Add('AK1,AK2').Pattern do
    begin
        Type := Integer(EXORGCHARTLib_TLB.exPatternFrame) Or
Integer(EXORGCHARTLib_TLB.exPatternFDiagonal);
        Color := $bebebe;
        FrameColor := $808080;
    end;
    EndUpdate();
end
```

VFP

```
with thisform.ChartView1
    .BeginUpdate
    with .Root
```

```
      .AddAssistant("Assistant 1")
      .AddAssistant("Assistant 2")
   endwith
   with .Nodes
      .Add("Child <b>1</b>",Null,"1234")
      .Add("Sub 1","1234","AK1")
      .Add("Sub 2","1234","AK2")
      .Add("Sub 3","1234")
      .Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
   endwith
   with .Frames.Add("AK1,AK2").Pattern
      .Type = 261 && PatternEnum.exPatternFrame Or
PatternEnum.exPatternFDiagonal
      .Color = RGB(190,190,190)
      .FrameColor = RGB(128,128,128)
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
local oChartView,var_Node,var_Nodes,var_Pattern

oChartView = form.EXORGCHARTACTIVEXCONTROL1.nativeObject
oChartView.BeginUpdate()
var_Node = oChartView.Root
   var_Node.AddAssistant("Assistant 1")
   var_Node.AddAssistant("Assistant 2")
var_Nodes = oChartView.Nodes
   var_Nodes.Add("Child <b>1</b>",null,"1234")
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Pattern = oChartView.Frames.Add("AK1,AK2").Pattern
   var_Pattern.Type = 261 /*exPatternFrame | exPatternFDiagonal*/
   var_Pattern.Color = 0xbebebe
```

```
   var_Pattern.FrameColor = 0x808080
oChartView.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oChartView as P
Dim var_Node as P
Dim var_Nodes as P
Dim var_Pattern as P

oChartView = topparent:CONTROL_ACTIVEX1.activex
oChartView.BeginUpdate()
var_Node = oChartView.Root
   var_Node.AddAssistant("Assistant 1")
   var_Node.AddAssistant("Assistant 2")
var_Nodes = oChartView.Nodes
   var_Nodes.Add("Child <b>1</b>",,"1234")
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Pattern = oChartView.Frames.Add("AK1,AK2").Pattern
   var_Pattern.Type = 261 'exPatternFrame + exPatternFDiagonal
   var_Pattern.Color = 12500670
   var_Pattern.FrameColor = 8421504
oChartView.EndUpdate()
```

Visual Objects

```
local var_Node as INode
local var_Nodes as INodes
local var_Pattern as IPattern

oDCOCX_Exontrol1:BeginUpdate()
var_Node := oDCOCX_Exontrol1:Root
   var_Node:AddAssistant("Assistant 1",nil,nil)
```

```
   var_Node:AddAssistant("Assistant 2",nil,nil)
var_Nodes := oDCOCX_Exontrol1:Nodes
   var_Nodes:Add("Child <b>1</b>",nil,"1234",nil,nil)
   var_Nodes:Add("Sub 1","1234","AK1",nil,nil)
   var_Nodes:Add("Sub 2","1234","AK2",nil,nil)
   var_Nodes:Add("Sub 3","1234",nil,nil,nil)
   var_Nodes:Add("Child <b>2.1</b>",nil,nil,nil,nil):AddGroup("Child
<b>2.2</b>",nil,nil)
var_Pattern := oDCOCX_Exontrol1:Frames:Add("AK1,AK2"):Pattern
   var_Pattern:Type := exPatternFrame | exPatternFDiagonal
   var_Pattern:Color := RGB(190,190,190)
   var_Pattern:FrameColor := RGB(128,128,128)
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oChartView,var_Node,var_Nodes,var_Pattern

oChartView = ole_1.Object
oChartView.BeginUpdate()
var_Node = oChartView.Root
   var_Node.AddAssistant("Assistant 1")
   var_Node.AddAssistant("Assistant 2")
var_Nodes = oChartView.Nodes
   var_Nodes.Add("Child <b>1</b>",,"1234")
   var_Nodes.Add("Sub 1","1234","AK1")
   var_Nodes.Add("Sub 2","1234","AK2")
   var_Nodes.Add("Sub 3","1234")
   var_Nodes.Add("Child <b>2.1</b>").AddGroup("Child <b>2.2</b>")
var_Pattern = oChartView.Frames.Add("AK1,AK2").Pattern
   var_Pattern.Type = 261 /*exPatternFrame | exPatternFDiagonal*/
   var_Pattern.Color = RGB(190,190,190)
   var_Pattern.FrameColor = RGB(128,128,128)
oChartView.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Variant voNode
   Get ComRoot to voNode
   Handle hoNode
   Get Create (RefClass(cComNode)) to hoNode
   Set pvComObject of hoNode to voNode
      Get ComAddAssistant of hoNode "Assistant 1" Nothing Nothing to Nothing
      Get ComAddAssistant of hoNode "Assistant 2" Nothing Nothing to Nothing
   Send Destroy to hoNode
   Variant voNodes
   Get ComNodes to voNodes
   Handle hoNodes
   Get Create (RefClass(cComNodes)) to hoNodes
   Set pvComObject of hoNodes to voNodes
      Get ComAdd of hoNodes "Child <b>1</b>" "1234" Nothing Nothing to
Nothing
      Get ComAdd of hoNodes "Sub 1" "1234" "AK1" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "Sub 2" "1234" "AK2" Nothing Nothing to Nothing
      Get ComAdd of hoNodes "Sub 3" "1234" Nothing Nothing Nothing to Nothing
      Variant voNode1
      Get ComAdd of hoNodes "Child <b>2.1</b>" Nothing Nothing Nothing
Nothing to voNode1
      Handle hoNode1
      Get Create (RefClass(cComNode)) to hoNode1
      Set pvComObject of hoNode1 to voNode1
         Get ComAddGroup of hoNode1 "Child <b>2.2</b>" Nothing Nothing to
Nothing
      Send Destroy to hoNode1
   Send Destroy to hoNodes
   Variant voFrames
   Get ComFrames to voFrames
   Handle hoFrames
   Get Create (RefClass(cComFrames)) to hoFrames
   Set pvComObject of hoFrames to voFrames
      Variant voFrame
```

```
      Get ComAdd of hoFrames "AK1,AK2" to voFrame
      Handle hoFrame
      Get Create (RefClass(cComFrame)) to hoFrame
      Set pvComObject of hoFrame to voFrame
         Variant voPattern
         Get ComPattern of hoFrame to voPattern
         Handle hoPattern
         Get Create (RefClass(cComPattern)) to hoPattern
         Set pvComObject of hoPattern to voPattern
            Set ComType of hoPattern to (OLEexPatternFrame +
OLEexPatternFDiagonal)
            Set ComColor of hoPattern to (RGB(190,190,190))
            Set ComFrameColor of hoPattern to (RGB(128,128,128))
         Send Destroy to hoPattern
      Send Destroy to hoFrame
   Send Destroy to hoFrames
   Send ComEndUpdate
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oChartView
   LOCAL oNode
   LOCAL oNodes
   LOCAL oPattern

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

```
   oChartView := XbpActiveXControl():new( oForm:drawingArea )
   oChartView:CLSID  := "Exontrol.ChartView.1" /*{F4DFE455-01FE-420E-A088-
64346DCC3791}*/
   oChartView:create(,, {10,60},{610,370} )

      oChartView:BeginUpdate()
      oNode := oChartView:Root()
         oNode:AddAssistant("Assistant 1")
         oNode:AddAssistant("Assistant 2")
      oNodes := oChartView:Nodes()
         oNodes:Add("Child <b>1</b>",,"1234")
         oNodes:Add("Sub 1","1234","AK1")
         oNodes:Add("Sub 2","1234","AK2")
         oNodes:Add("Sub 3","1234")
         oNodes:Add("Child <b>2.1</b>"):AddGroup("Child <b>2.2</b>")
      oPattern := oChartView:Frames():Add("AK1,AK2"):Pattern()
         oPattern:Type := 261/*exPatternFrame+exPatternFDiagonal*/
         oPattern:SetProperty("Color",AutomationTranslateColor( GraMakeRGBColor  (
{ 190,190,190 } )  , .F. ))
         oPattern:SetProperty("FrameColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 128,128,128 } )  , .F. ))
      oChartView:EndUpdate()

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

method Frames.Clear ()

Removes all objects in a collection.

Type	Description

The Clear method removes all Frame objects. The Remove method removes a specified frame. The Visible property shows or hides a specified frame.

property Frames.Count as Long

Returns the number of elements in the collection.

Type	Description
Long	A Long expression that indicates the number of Frame objects within the chart.

The Count property returns the number of Frame objects in the collection. The Clear method removes all Frame objects. The Remove method removes a specified frame. The Visible property shows or hides a specified frame. You can use the Item and Count properties to enumerate the frames in the control, as well as **for each** statement.

property Frames.Item (Index as Variant) as Frame

Returns a specific Frame of the Frames collection, giving its index.

Type	Description
Index as Variant	A Long expression that defines the index of the frame to be requested.
Frame	A Frame object to be requested.

The Item property retrieves a specified frame object. You can use the Item and Count properties to enumerate the frames in the control, as well as **for each** statement. The Count property returns the number of Frame objects in the collection. The Clear method removes all Frame objects. The Remove method removes a specified frame. The Visible property shows or hides a specified frame.

method Frames.Remove (Index as Variant)

Removes a specific member from the Frames collection, giving its index or reference.

Type	Description
Index as Variant	A Long expression that defines the index of the frame to be deleted.

The Remove method removes a specified frame. The Clear method removes all Frame objects. The Visible property shows or hides a specified frame.

Node object

The Node object holds information about a node into the chart. Use the Nodes property to access the nodes collection. The Node object supports the following properties:

Name	Description
AddAssistant	Adds an assistant node.
AddGroup	Adds a node in the same group.
Alignment	Specifies the alignment of the caption within the node.
ArrangeSiblingNodesAs	Specifies whether the first child node and its siblings nodes are arranged vertically or horizontally.
Assistant	Retrieves an assistant node by its index.
BackColor	Retrieves or sets a value that indicates the node's background color.
BackgroundExt	Indicates additional colors, text, images that can be displayed on the node's background using the EBN string format.
BackgroundExtValue	Specifies at runtime, the value of the giving property for specified part of the background extension.
BorderColor	Specifies the border's color.
BorderWidth	Specifies the width of the border.
Caption	Specifies the node's caption.
CaptionSingleLine	Specifies if the element's caption is displayed on single or multiple lines.
ClearAssistants	Clears the assistant nodes.
ClearBackColor	Clears the node's background color.
ClearForeColor	Clears the node's foreground color.
ClearGroup	Clears the nodes in the same group.
CountAssistants	Specifies the number of assistant nodes.
CountGroup	Specifies the number of nodes in the same group.
DrawRoundNode	Specifies a value that indicates whether the node has borders with round corners.
Editable	Specifies whether the node's caption is editable.
Enabled	Enables or disables the node.
Expanded	Sets or returns whether a node in the hierarchy is

	expanded.
FirstNode	Gets the first child node in the node collection.
FixedHeight	Retrieves or sets a value that indicates whether the height of the node's caption is fixed.
FixedWidth	Retrieves or sets a value that indicates whether the width of the node's caption is fixed.
ForeColor	Retrieves or sets a value that indicates the node's foreground color.
Group	Retrieves an node in the group by its index.
HasButton	Specifies whether the node displays the +/- buttons.
Image	Specifies a value that indicates the index of image being used.
ImageAlignment	Specifies the alignment of the image within the node.
Index	Gets the index of the node within the control Nodes collection.
InflateGroupX	Increases the width of the group.
InflateGroupY	Increases the height of the group.
IsAssistant	Retrieves a value that specifies whether the node is an assistant.
IsGroup	Retrieves a value that specifies whether the node belongs to a group.
Key	Specifies the node's key.
LastNode	Gets the last child node.
Left	Retrieves or sets a value that indicates whether the assistant node is on the left side.
LinkCaption	Specifies the caption on the node's link.
LinkTo	Retrieves or sets a value that indicates the list of nodes that the source node links to.
LinkToCaption	Specifies the HTML caption being shown on a LinkTo line.
LinkToColor	Specifies the color to show the LinkTo line.
LinkToPen	Specifies the style of the link for linkto line.
LinkToRound	Specifies whether the LinkTo line is shown linear or round.
LinkToShowDir	Specifies whether the LinkTo line shows the direction.
LinkToWidth	Specifies the width to display the LinkTo line.

NextNode	Gets the next sibling node.
NodeCount	Retrieves the number of child nodes.
Nodes	Gets the collection of nodes.
Padding	Returns or sets a value that indicates the padding of the node.
Parent	Retrieves or sets the parent node
PenBorderNode	Specifies the type of pen used to paint the node's borders.
Picture	Retrieves or sets a graphic to be displayed in the node.
PictureAlignment	Specifies the alignment of the picture within the node.
PictureAspectRatio	Specifies the aspect ratio of the node's picture.
PictureHeight	Specifies the height of the node's picture.
PictureWidth	Specifies the width of the node's picture.
Position	Specifies the position of the node.
PrevNode	Gets the previous sibling node.
Remove	Removes recursively the child nodes, the assistant nodes and all the nodes in the same group, and if possible remove the node itself.
RemoveAssistant	Removes an assistant node.
RemoveGroup	Removes a node from the group.
ShadowNode	Specifies whether the node has shadow.
ShowLinkDir	Shows or hides the direction between the node and its parent.
ShowLinks	Shows or hides the links between node and its child nodes or its parent, if is it an assistant node.
ShowRoundLink	Specifies whether the round links are shown between parent and child nodes.
ToolTip	Specifies the description for the node's tooltip.
ToolTipTitle	Specifies the title of the node's tooltip.
UserData	Assigns an user extra data to the node.
Width	Specifies the maximum width of the node's caption.

method Node.AddAssistant (Caption as String, [Image as Variant], [Picture as Variant])

Adds an assistant node.

Type	Description
Caption as String	A string expression that identifies the node's caption. The Caption supports built-in HTML tags.
Image as Variant	A long expression that indicates the index of the icon being assigned to the node.
Picture as Variant	A string expression that indicates the path to the picture file being loaded by the node, a IPictureDisp object that specifies the picture of the node

Return	Description
Node	A Node object being created.

Use the AddAssistant method to add assistant nodes. An assistant node is displayed between a child node and its parent. The Node being returned by the AddAssistant method has the IsAssistant property on True. The control displays the assistant nodes only if the ShowAssistants property is True. An assistant node is displayed only if the parent node contains child nodes. By default, the control aligns the assistant nodes like follows: first assistant node is aligned to the left, the second assistant node is aligned to the right, the third assistant node to the left and so on. The Left property aligns the node to the left or right side of the link between child nodes and their parent. Use the Caption property to change the node's caption at runtime. Use the Assistant property to access the assistant nodes collection. Use the RemoveAssistant method to remove an assistant node. Use the PenLinkAssistant property to define the type of the pen used to paint the links between assistant nodes. *Please not that an assistant node is shown between parent and child node, so if the node is not expanded, or has no childs, the assistant nodes are not shown.*

The Caption parameter supports the following built-in HTML tags:

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

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

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=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

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

gradient-center

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

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

outlined

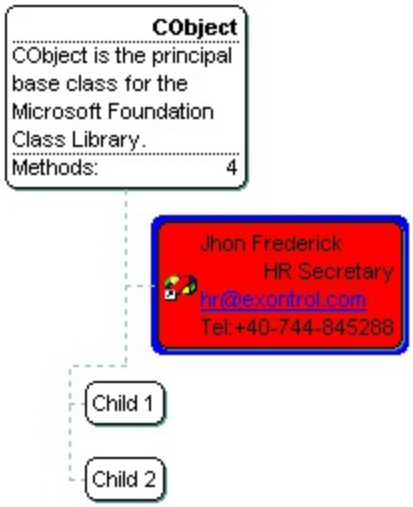
- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2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

shadow

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

outline anti-aliasing

In the following image, the red node represents an assistant node:



The following VB sample adds an assistant node to the root object:

```
Private Sub Form_Load()
  With ChartView1
    .BeginUpdate
    With .Root
      .Caption = "<r><dotline><b>CObject</b><br>CObject is the principal base
```

```
class for the Microsoft Foundation Class Library.<br><upline><dotline>Methods:<r>4"
        With .AddAssistant("Jhon Frederick<br><r>HR Secretary<br><fgcolor=0000FF>
<u>hr@exontrol.com</u></fgcolor><br>Tel:<r>+40-744-845288", 1)
            Left = False
        End With
    End With
    With .Nodes
        .Add "Child 1"
        .Add "Child 2"
    End With
    .EndUpdate
  End With
End Sub
```

The following C++ sample adds an assistant node to the root object:

```
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CNode node = m_chartview.GetRoot();
CNode assistant = node.AddAssistant( "New Assistant", vtMissing, vtMissing );
assistant.SetBackColor( RGB(0,0,255) );
assistant.SetForeColor( RGB(255,255,255) );
```

The following VB.NET sample adds an assistant node to the root object:

```
With AxChartView1.Root
  With .AddAssistant("New Assistant")
    .ForeColor = ToUInt32(Color.White)
    .BackColor = ToUInt32(Color.Blue)
  End With
End With
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
  Dim i As Long
  i = c.R
  i = i + 256 * c.G
  i = i + 256 * 256 * c.B
  ToUInt32 = Convert.ToUInt32(i)
```

```
End Function
```

The following C# sample adds an assistant node to the root object:

```
EXORGCHARTLib.Node node = axChartView1.Root.AddAssistant("New Assistant", null,
null);
node.ForeColor = ToUInt32(Color.White);
node.BackColor = ToUInt32(Color.Blue);
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample adds an assistant node to the root object:

```
with thisform.ChartView1.Root
   with .AddAssistant( "New Assistant" )
      .ForeColor = RGB(255,255,255)
      .BackColor = RGB(0,0,255)
   endwith
endwith
```

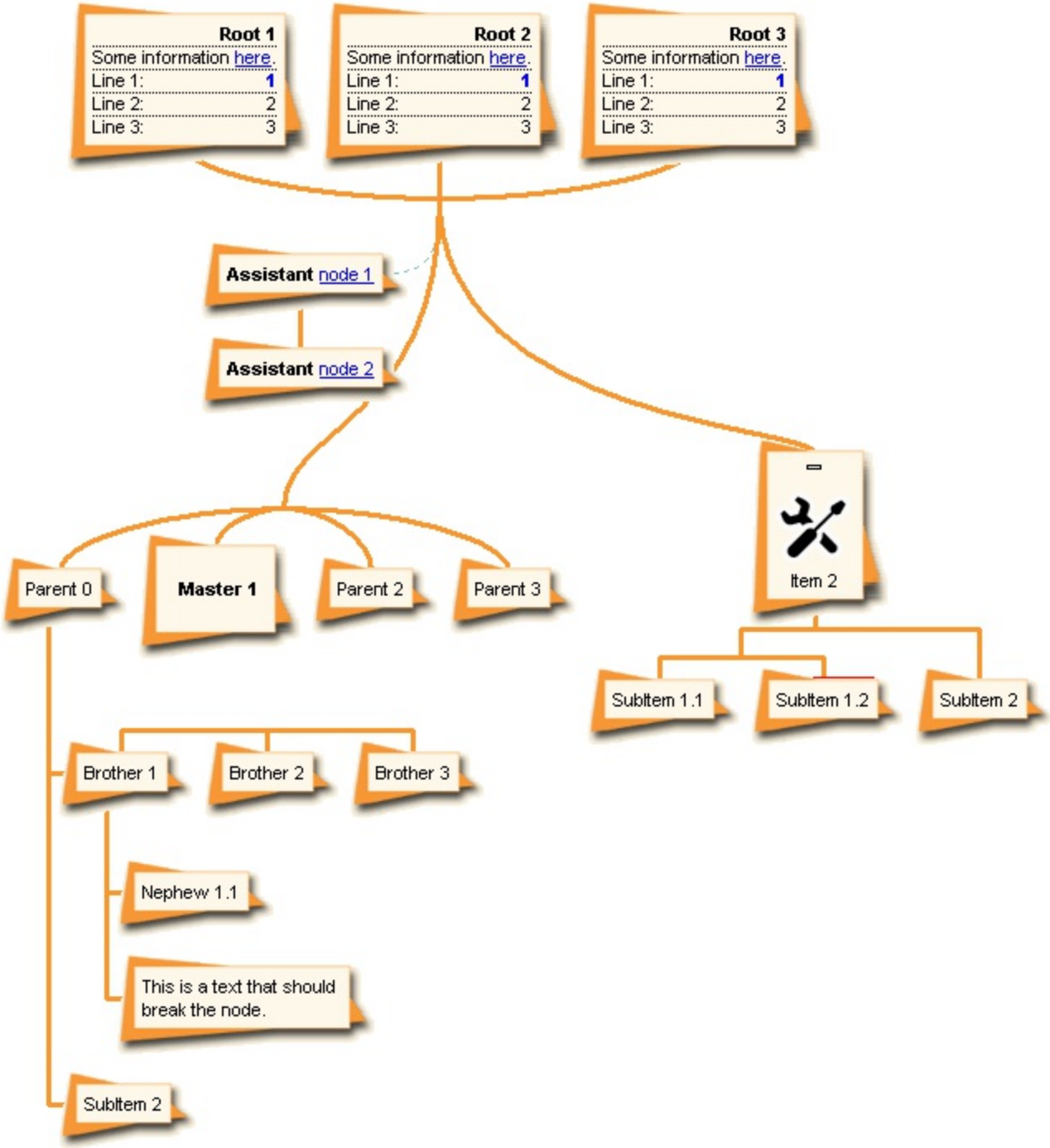
method Node.AddGroup (Caption as String, [Image as Variant], [Picture as Variant])

Adds a node in the same group.

Type	Description
Caption as String	A String expression that indicates the caption of the node being added. The Caption supports built-in HTML tags.
Image as Variant	long expression that indicates the index of the icon being assigned to the node.
Picture as Variant	A string expression that indicates the path to the picture file being loaded by the node, a IPictureDisp object that specifies the picture of the node

Return	Description
Node	A Node object being created.

Use the AddGroup method to add new nodes in the same group in other words it allows you to add multiple parents. For instance, you can add new root objects by using the Root.AddGroup method. The Root2 and Root3 nodes in the following screen shot were added using the Root.AddGroup. Use the CountGroup property to count the number of nodes in the same group that belongs to a node. Use the InflateGroupY property to specify the indentation of the group on vertical axis. Use the IsGroup property to specify whether a node was added using the AddGroup method.



The Caption parameter supports the following built-in HTML tags:

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

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=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).

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

gradient-center

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

height of the font. For instance the "<**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

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

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

property Node.Alignment as AlignmentEnum

Specifies the alignment of the caption within the node.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the node.

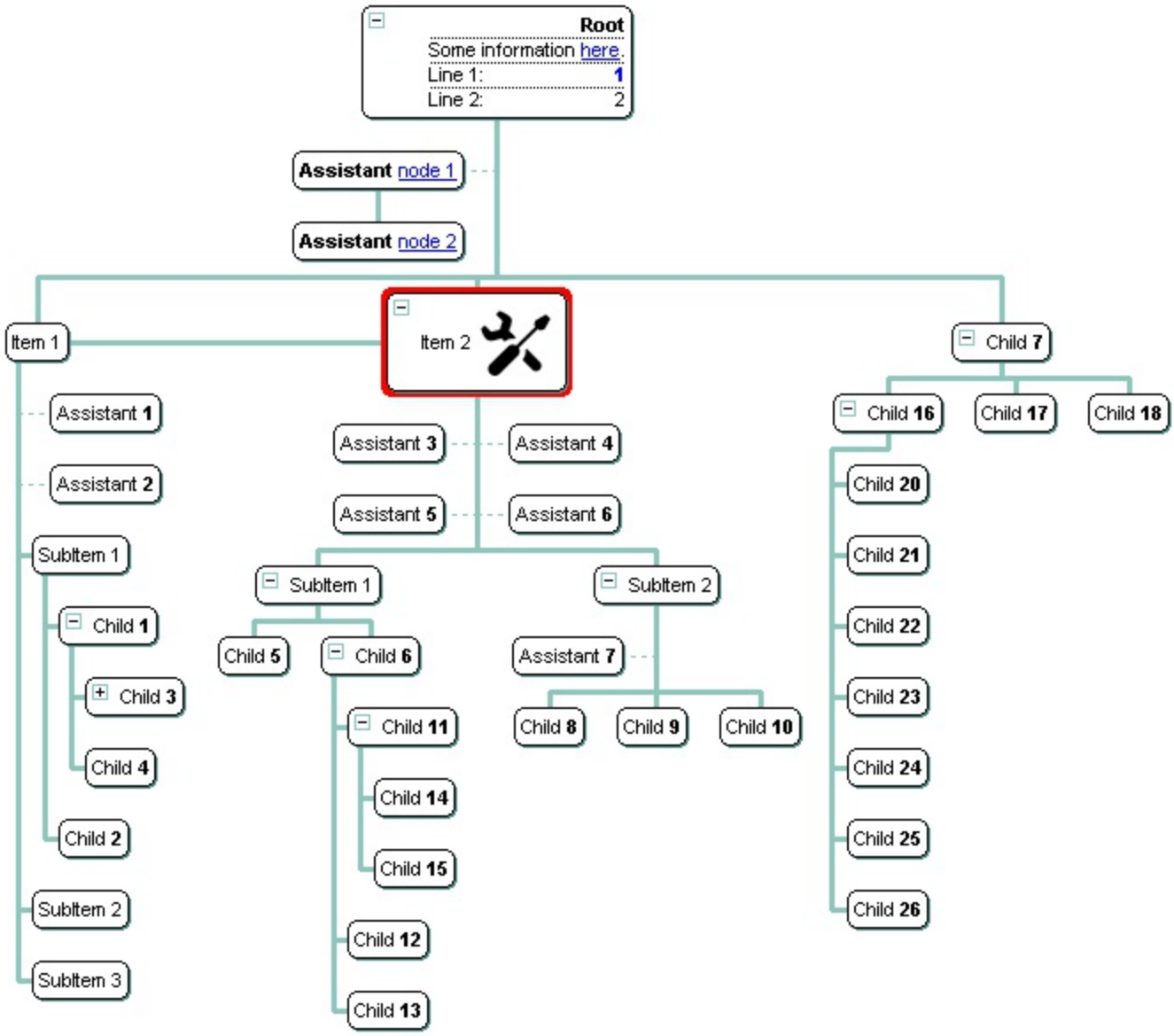
By default, the Alignment property is LeftAlignment. Use the FixedWidth property to specify the width of the node. Use the Alignment property to align the caption on the node.

property Node.ArrangeSiblingNodesAs as ArrangeSiblingEnum

Specifies whether the first child node and its siblings nodes are arranged vertically or horizontally.

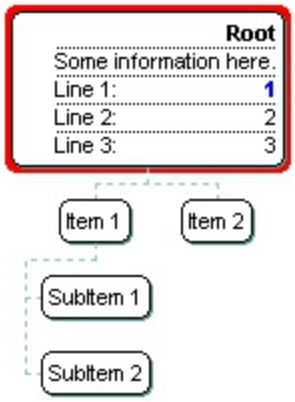
Type	Description
ArrangeSiblingEnum	An ArrangeSiblingEnum expression that indicates whether the first child node and its siblings nodes are arranged vertically or horizontally.

By default, the ArrangeSiblingNodesAs property is exDefault Use the ArrangeSiblingNodesAs property to align horizontally or as tree the child nodes. Use the IndentSiblingX property to indent horizontally the sibling nodes. Use the IndentSiblingY property to indent vertically the sibling nodes. The IndentChild property retrieves or sets the amount, in pixels, that child nodes are indented relative to their parent nodes, which has effect if the ArrangeSiglingNodesAs property is exTree. *Please notice that the assistant node is always aligned to the right, no matter of the Left property, if is a child of a node that has the ArrangeSiblingNodesAs property on exTree.*

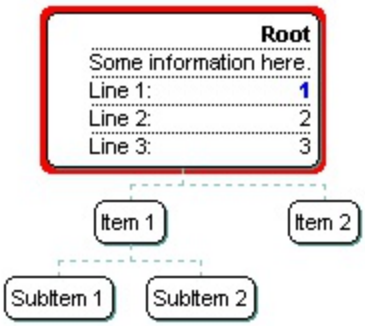


For instance, in the above screen shot, the **Item 1** has its child nodes as a tree, so the ArrangeSiblingNodesAs for this node is exTree. The **Child 7**, displays horizontally, its child nodes from left to right, so its ArrangeSiblingNodesAs property is exHorizontally, and the **Child 16** arranges its child nodes vertically from top to bottom, so its ArrangeSiblingNodesAs property is exDefault.

The following screen shot shows how the child nodes are arranged when the ArrangeSiblingNodesAs property is exDefault (see the child nodes for the **"Item 1"** node) :



The following screen shot shows how the child nodes are arranged when the ArrangeSiblingNodesAs property is exHorizontally (see the child nodes for the **"Item 1"** node) :



property Node.Assistant (Index as Variant) as Node

Retrieves an assistant node by its index.

Type	Description
Index as Variant	A long expression that indicates the index of assistant node being accessed
Node	An assistant Node being accessed.

Use the Assistant property to access to the assistant nodes collection. Use the CountAssistants property to get the number of the assistant nodes assigned to the node. Use the AddAssistant property to add an assistant node. Use the RemoveAssistant method to remove an assistant node. Use the IsAssistant property to specify whether the node is an assistant node or a child node.

The following VB sample enumerates the assistant nodes, for the root node:

```
With ChartView1.Root
    For i = 0 To .CountAssistants - 1
      Debug.Print .Assistant(i).Caption
    Next
End With
```

The following C++ sample enumerates the assistant nodes, for the root node:

```
CNode root = m_chartview.GetRoot();
for ( long i = 0; i < root.GetCountAssistants(); i++ )
{
  CNode assistant = root.GetAssistant( COleVariant( i ) );
  OutputDebugString( assistant.GetCaption() );
}
```

The following VB.NET sample enumerates the assistant nodes, for the root node:

```
With AxChartView1.Root
  Dim i As Integer
  For i = 0 To .CountAssistants - 1
    Debug.WriteLine(.Assistant(i).Caption())
  Next
End With
```

The following C# sample enumerates the assistant nodes, for the root node:

```
EXORGCHARTLib.Node root = axChartView1.Root;
for (int i = 0; i < root.CountAssistants; i++)
{
   EXORGCHARTLib.Node node = root.get_Assistant(i);
   System.Diagnostics.Debug.WriteLine(node.Caption);
}
```

The following VFP sample enumerates the assistant nodes, for the root node:

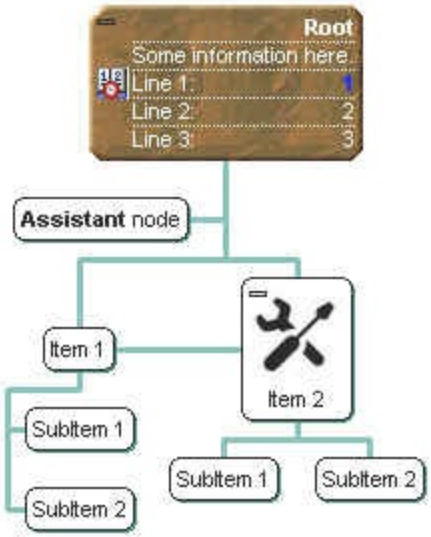
```
with thisform.ChartView1.Root
   For i = 0 To .CountAssistants - 1
      wait window  nowait .Assistant(i).Caption
   Next
endwith
```

property Node.BackColor as Color

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

Type	Description
Color	A color expression that defines the node'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 BackColor property to specify the node's background color. Use the BackColorNode property to define the default background color for nodes. Use the ClearBackColor method to clear the node's background color. Use the BackColor property to specify the control's background. Use the SelColor property to change the visual appearance for the selected node. The Background property specifies a background color or a visual appearance for specific parts in the control. Use the BorderColor property to assign a different color for the node's border. The BackgroundExt property provides unlimited options to add more colors, patterns, text, icons, pictures, frames to any node. Use the Padding property of the Node to define the padding for specified node.



The following VB sample changes the visual appearance for the root node. The sample uses the "  " skin.

```
With ChartView1
  .VisualAppearance.Add 3, "D:\Temp\ExOrgChart.Help\button.ebn"
  With .Root
    .ForeColor = RGB(255, 255, 255)
```

```
        .BackColor = &H3000000
    End With
End With
```

The following C++ sample changes the visual appearance for the root node:

```
#include "Appearance.h"
#include "Node.h"
m_chartview.GetVisualAppearance().Add( 3,
COleVariant("D:\\Temp\\ExOrgChart.Help\\button.ebn") );
CNode node = m_chartview.GetRoot();
node.SetForeColor( RGB(255,255,255) );
node.SetBackColor( 0x3000000 );
```

The following VB.NET sample changes the visual appearance for the root node:

```
With AxChartView1
    .VisualAppearance.Add(3, "D:\Temp\ExOrgChart.Help\button.ebn")
    With .Root
        .ForeColor = RGB(255, 255, 255)
        .BackColor = &H3000000
    End With
End With
```

The following C# sample changes the visual appearance for the root node:

```
axChartView1.VisualAppearance.Add(3, "D:\\Temp\\ExOrgChart.Help\\button.ebn");
EXORGCHARTLib.Node node = axChartView1.Root;
node.ForeColor = ToUInt32(Color.FromArgb(255, 255, 255));
node.BackColor = 0x3000000;
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR expression:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
```

```
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the visual appearance for the root node:

```
With thisform.ChartView1
    .VisualAppearance.Add(3, "D:\Temp\ExOrgChart.Help\button.ebn")
    with .Root
        .ForeColor = RGB(255,255,255)
        .BackColor = 50331648
    endwith
EndWith
```

where the 33554432 in hexa is 0x2000000.

The following VB sample changes the background and foreground color for the selected node:

```
Private Sub ChartView1_Select(ByVal OldNode As EXORGCHARTLibCtl.INode, ByVal
NewNode As EXORGCHARTLibCtl.INode)
    If Not (OldNode Is Nothing) Then
        With OldNode
            .ClearBackColor
            .ClearForeColor
        End With
    End If
    With NewNode
        .ForeColor = vbWhite
        .BackColor = vbBlue
    End With
End Sub
```

The following C++ sample changes the background and foreground color for the selected node:

```
void OnSelectChartview1(LPDISPATCH OldNode, LPDISPATCH NewNode)
{
    CNode oldNode( OldNode ); oldNode.m_bAutoRelease = FALSE;
    CNode newNode( NewNode ); newNode.m_bAutoRelease = FALSE;
```

```
    if ( oldNode.m_lpDispatch != NULL )
    {
        oldNode.ClearBackColor();
        oldNode.ClearForeColor();
    }
    newNode.SetBackColor( RGB(0,0,128) );
    newNode.SetForeColor( RGB(255,255,255) );
}
```

The following VB.NET sample changes the background and foreground color for the selected node:

```
Private Sub AxChartView1_SelectEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_SelectEvent) Handles AxChartView1.SelectEvent
    If Not (e.oldNode Is Nothing) Then
        With e.oldNode
            .ClearBackColor()
            .ClearForeColor()
        End With
    End If
    With e.newNode
        .ForeColor = ToUInt32(Color.White)
        .BackColor = ToUInt32(Color.Blue)
    End With
End Sub
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background and foreground color for the selected node:

```
private void axChartView1_SelectEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_SelectEvent e)
{
   if ( e.oldNode != null )
   {
      e.oldNode.ClearBackColor();
      e.oldNode.ClearForeColor();
   }
   e.newNode.BackColor = ToUInt32(Color.Blue);
   e.newNode.ForeColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background and foreground color for the selected node:

```
*** ActiveX Control Event ***
LPARAMETERS oldnode, newnode

If !isnull(oldnode)
   With oldnode
      .ClearBackColor
      .ClearForeColor
   EndWith
EndIf
With newnode
   .ForeColor = RGB(255,255,255)
   .BackColor = RGB(0,0,128)
```

EndWith

property Node.BackgroundExt as String

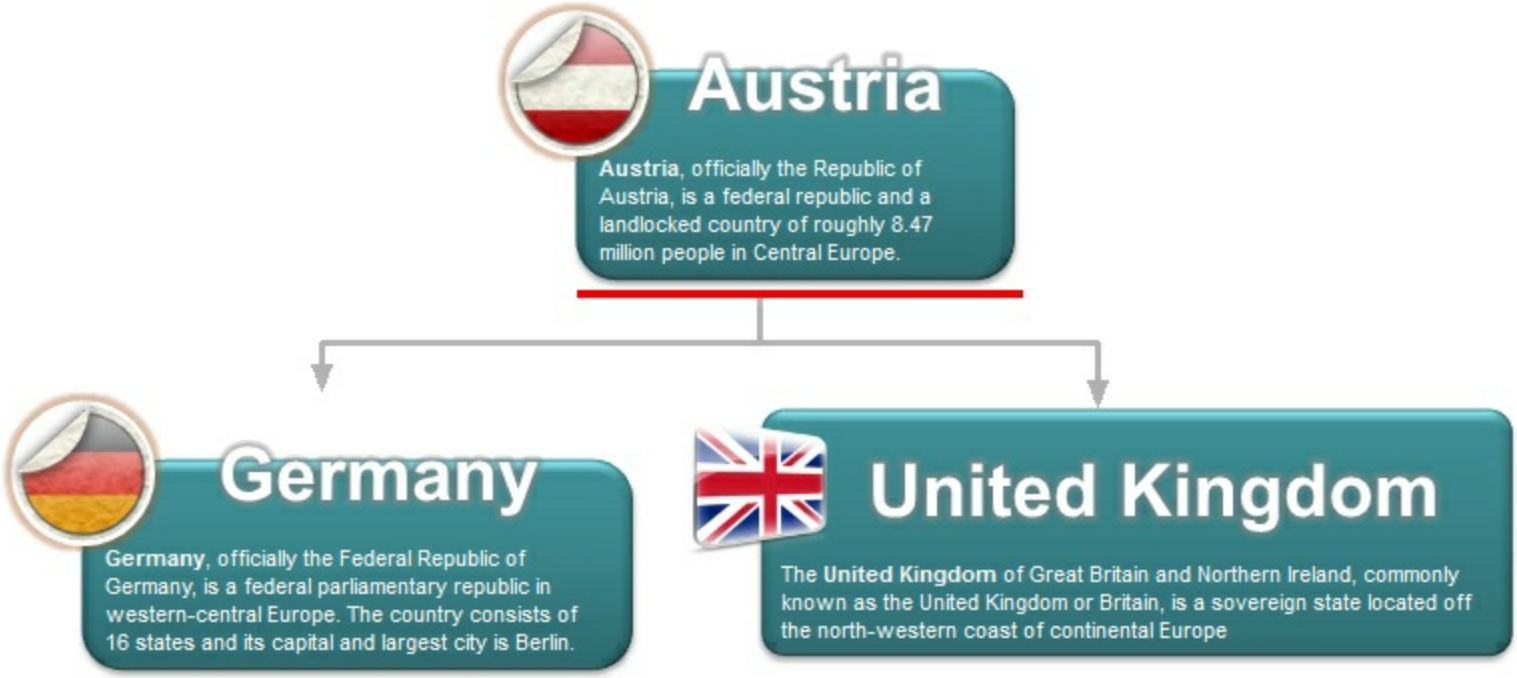
Indicates additional colors, text, images that can be displayed on the object's background using the EBN string format.

Type	Description
String	A String expression ("**EBN String Format**") that defines the layout of the UI to be applied on the object's background. The syntax of EBN String Format in BNF notation is shown bellow. *You can use the EBN's Builder of eXButton/COM control to define visually the EBN String Format.*

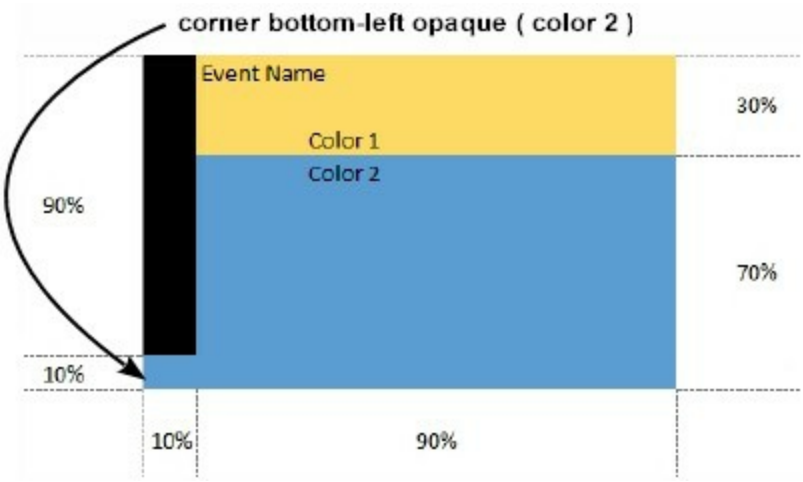
By default, the BackgroundExt property is empty. Using the BackgroundExt property you have unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the object's background. *For instance, let's say you need to display **more** colors on the object's background, or just want to display an **additional** caption or image to a specified location on the object's background.* The EBN String Format defines the parts of the EBN to be applied on the object's background. The EBN is a set of UI elements that are built as a tree where each element is anchored to its parent element. Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExt property is applied right after setting the object's backcolor, and before drawing the default object's captions, icons or pictures. Use the Padding property of the Node to define the padding for specified node.

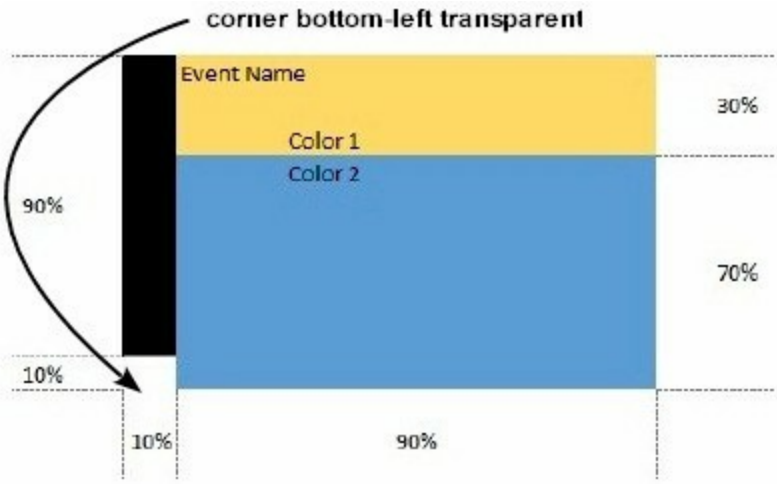
The following screen shot shows how you can extend the node as follows:

- displays the picture to a different place
- assign more HTML captions to the node
- different type of borders/frames
- and so on.



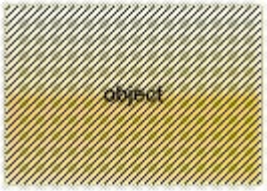
Complex samples:



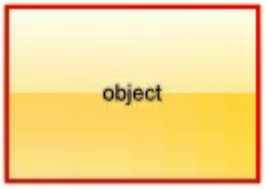


Easy samples:

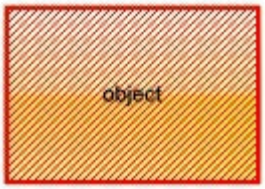
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



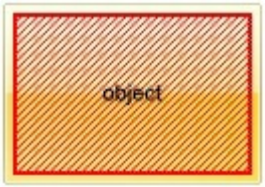
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



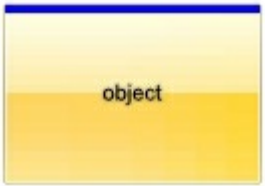
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



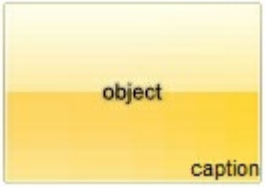
- "[[patterncolor=RGB(255,0,0)] (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.

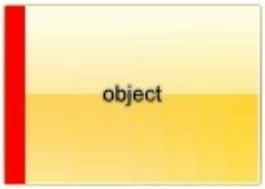


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

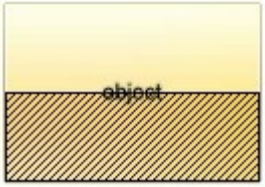


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

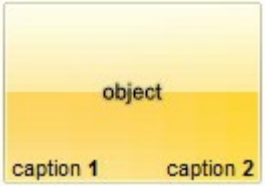
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



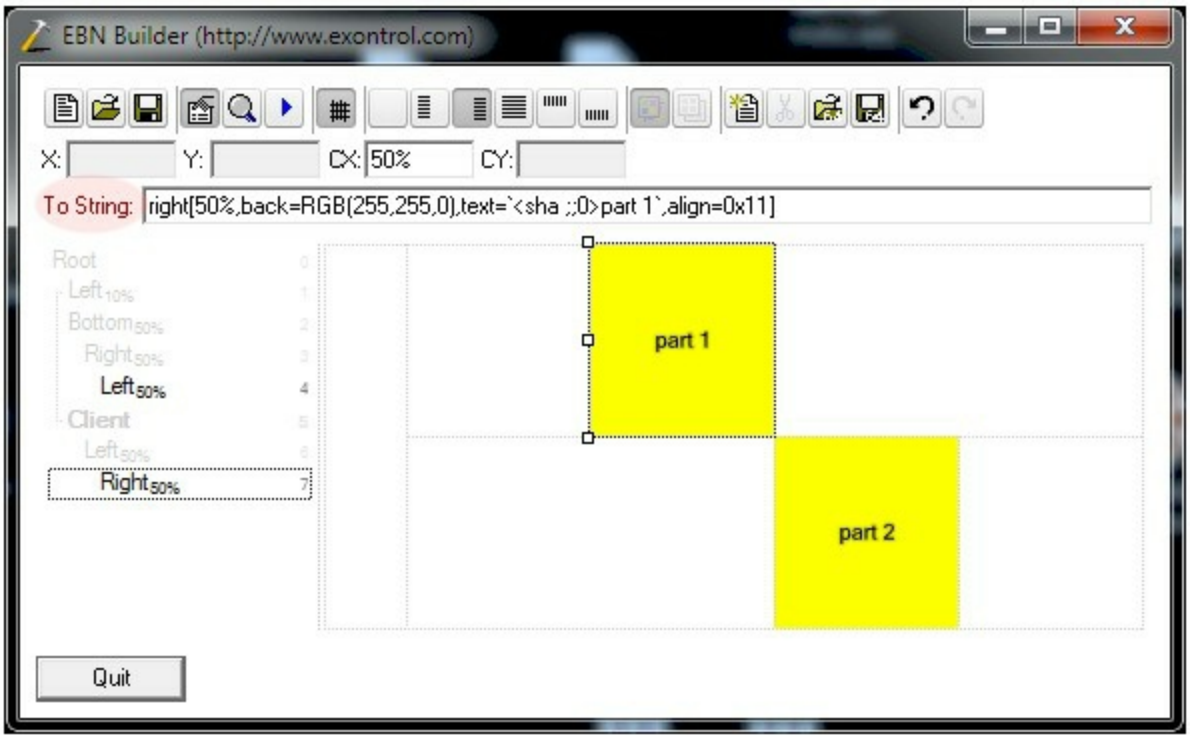
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



- "root[text=`caption <b>2`,align=0x22](client[text=`caption <b>1`,align=0x20])", shows the caption **1** aligned to the bottom-left side, and the caption **2** to the bottom-right side



*The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field as shown in the following screen shot:*



The **To String** field of the EBN Builder defines the **EBN String Format** that can be used on BackgroundExt property.

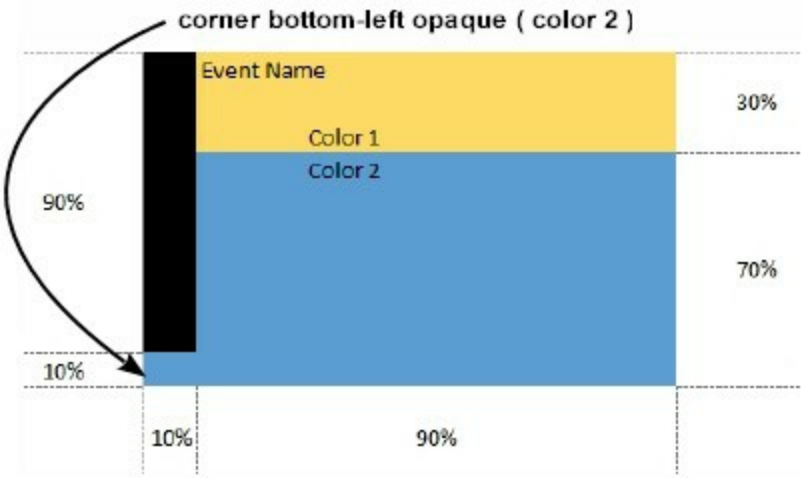
The **EBN String Format** syntax in BNF notation is defined like follows:

```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<client> ","] <attribute> [ "," <attributes> ] "]"
<client> ::= <expression> | <expression> "," <expression> "," <expression> ","
<expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> | <wordwrap> | <align> | <pattern> |
<patterncolor> | <frame> | <framethick> | <data> | <others>
<equal> ::= "="
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"  "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number> ")"
```

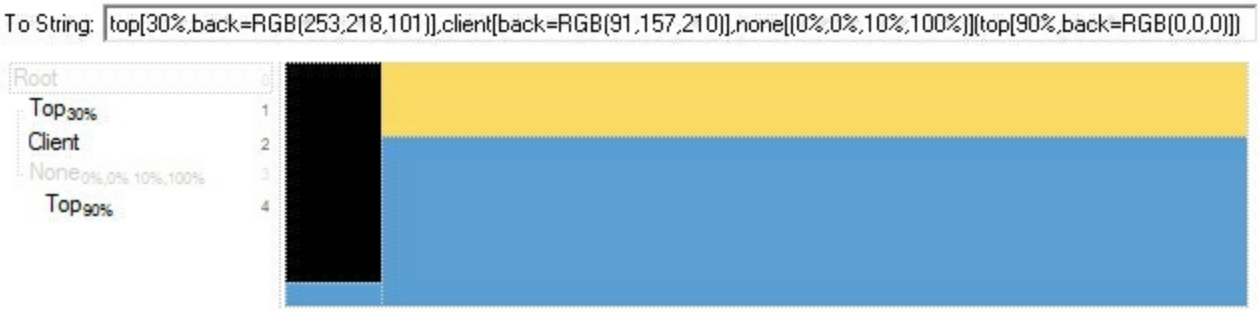
```
<string> ::= "`" <characters> "`" | "'" <characters> "'" | " <characters> "
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
<patterncolor> ::= "patterncolor" <equal> <color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

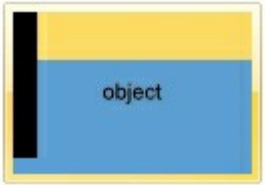
Now, lets say we have the following request to layout the colors on the objects:



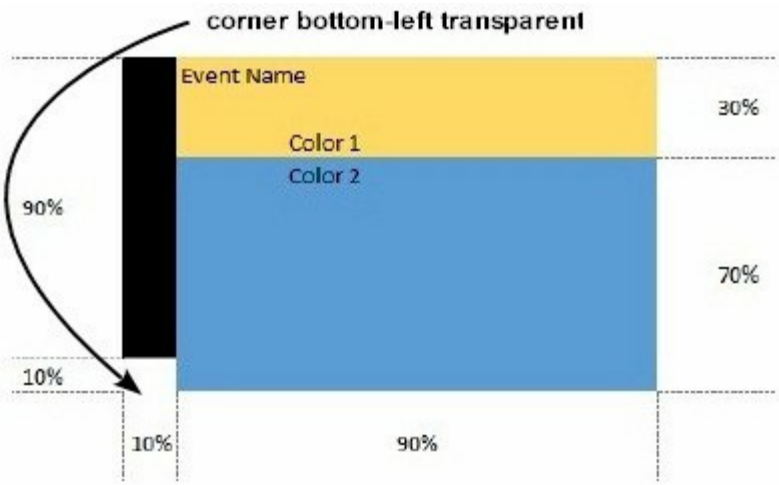
We define the BackgroundExt property such as "top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100%
(top[90%,back=RGB(0,0,0)])", and it looks as:



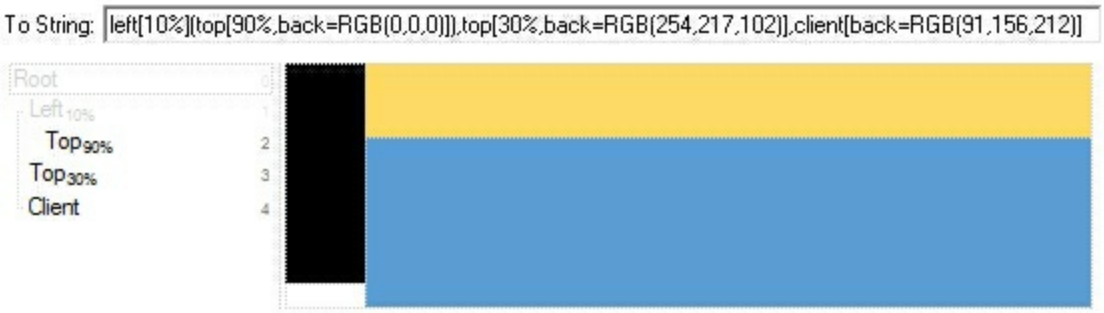
so, if we apply to our object we got:



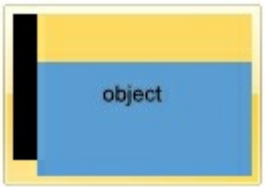
Now, lets say we have the following request to layout the colors on the objects:



We define BackgroundExt property such as "left[10%] (top[90%,back=RGB(0,0,0)]),top[30%,back=RGB(254,217,102)],client[back=RGB(91,156,2
and it looks as:

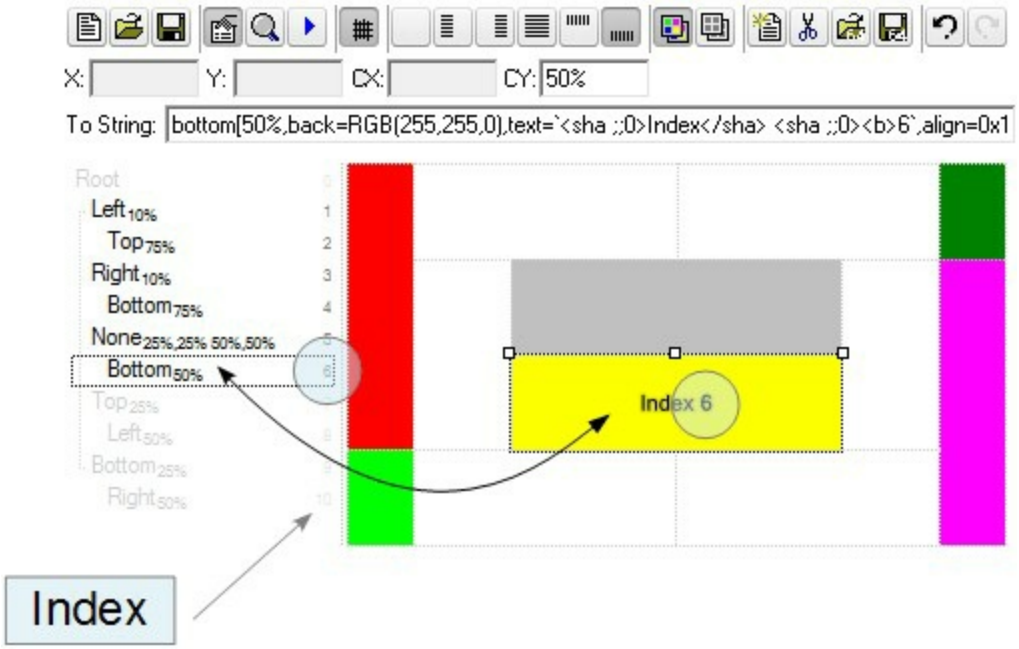


so, if we apply to our object we got:



property Node.BackgroundExtValue(Index as IndexExtEnum, Property as BackgroundExtPropertyEnum) as Variant

Specifies at runtime, the value of the giving property for specified part of the background extension.

Type	Description
Index as IndexExtEnum	A Long expression that defines the index of the part that composes the EBN to be accessed / changed. The following screen shot shows where you can find Index of the parts:  The screen shot shows that the EBN contains 11 elements, so in this case the Index starts at 0 (root element) and ends on 10.
Property as BackgroundExtPropertyEnum	A BackgroundExtPropertyEnum expression that specifies the property to be changed as explained bellow.
Variant	A Variant expression that defines the part's value. The Type of the expression depending on the Property parameter as explained bellow.

Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExtValue property has no effect if the BackgroundExt property is empty (by default). *The idea is as follows: first you need to decide the layout of the UI to put on the object's background, using the*

BodyBackgroundExt property, and next (if required), you can change any property of any part of the background extension to a new value. In other words, let's say you have the same layout to be applied to some of your objects, so you specify the BodyBackgroundExt to be the same for them, and next use the BackgroundExtValue property to change particular properties (like back-color, size, position, anchor) for different objects.

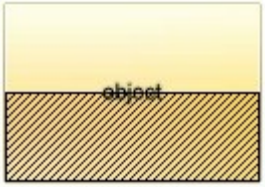
You can access/define/change the following UI properties of the element:

- **exBackColorExt**(1), Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000. (Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*
- **exClientExt**(2), Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect. Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent. (String/Numeric expression)*
- **exAnchorExt**(3), Specifies the object's alignment relative to its parent. *(Numeric expression)*
- **exTextExt**(4), Specifies the HTML text to be displayed on the object. *(String expression)*
- **exTextExtWordWrap**(5), Specifies that the object is wrapping the text. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *(Boolean expression)*
- **exTextExtAlignment**(6), Indicates the alignment of the text on the object. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag *(Numeric expression)*
- **exPatternExt**(7), Indicates the pattern to be shown on the object. The exPatternColorExt specifies the color to show the pattern. *(Numeric expression)*
- **exPatternColorExt**(8), Indicates the color to show the pattern on the object. The exPatternColorExt property has effect only if the exPatternExt property is not 0 (empty). The exFrameColorExt specifies the color to show the frame (the exPatternExt property includes the exFrame or exFrameThick flag). *(Color expression)*
- **exFrameColorExt**(9), Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. *(Color expression)*
- **exFrameThickExt**(11), Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property. *(Boolean expression)*
- **exUserDataExt**(12), Specifies an extra-data associated with the object. *(Variant*

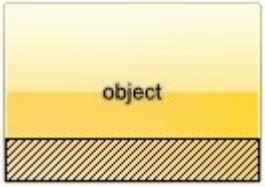
expression)

For instance, having the BodyBackgroundExt on "bottom[50%,pattern=6,frame]"

we got:



so let's change the percent of 50% to 25% like BackgroundExtValue(1,2) on "25%", where 1 indicates the first element after root, and 2 indicates the **exClientExt** property, we get:



In VB you should have the following syntax:

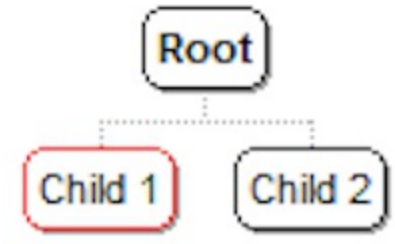
```
.BodyBackgroundExt = "bottom[50%,pattern=6,frame]"
.BackgroundExtValue(exIndexExt1, exClientExt) = "25%"
```

property Node.BorderColor as Color

Specifies the border's color.

Type	Description
Color	A Color expression that specifies the color to show the node's border.

By default, the BorderColor property is 0. Use the BorderColor property to assign a different color for the node's border. The BackColor property specifies the node's background color. The BorderWidth property indicates the width of the node's border. The ShadowNode property property shows or hides the shadow for a specific node. The ShadowNode property determines whether the control displays a shadow for nodes. Use the PenBorderNode property to define the type of the pen used to paint the borders for a specified node. Use the DrawRoundNode property to specify whether the node has a round border.

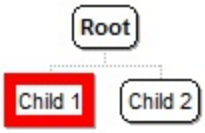


property Node.BorderWidth as Long

Specifies the width of the border.

Type	Description
Long	A Long expression that specifies the width of the node's border, in pixels.

By default, the BorderWidth property is 1. Use the BorderWidth property to assign a different size for the node's border. The BackColor property specifies the node's background color. The BorderWidth property indicates the width of the node's border. The ShadowNode property property shows or hides the shadow for a specific node. The ShadowNode property determines whether the control displays a shadow for nodes. Use the PenBorderNode property to define the type of the pen used to paint the borders for a specified node. Use the DrawRoundNode property to specify whether the node has a round border.



property Node.Caption as String

Specifies the node's caption.

Type	Description
String	A string expression that indicates the caption of the node.

Use the Caption property to specify the node's caption. Use the Image property to assign an icon to a node. Use the Picture property to assign a custom size picture to a node. Use the Root property to get the root node. Use the Add method to add a new child node. Use the BeginUpdate and EndUpdate methods to maintain performance while adding new nodes to the control. Use the **<img>** HTML tag to insert icons inside the node's caption. Use the LinkCaption property to specify a caption on the line that links the parent with the current node. Use the EditNode to programmatically edit the giving node.

The Caption property supports built-in HTML format. The supported tags are:

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

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

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

  gradient-center

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

  outlined

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

  shadow

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

You can define the node's caption when calling the Add method.

```
With ChartView1.Root
      .Caption = "<r><dotline><b><fgcolor=0000FF>Andrew Fuller</fgcolor></b>
<br><solidline><b>Title</b>:<r><fgcolor=FF0000>Vice President Sales</fgcolor>
<br>USA, Tacoma, WA, 98401, 908 W. Capital Way<br><dotline><upline><b>Phone:
</b><r>(206) 555-9482"
End With
```

The above sample looks like following:

The following C++ sample changes the caption for the root node:

```
CNode root = m_chartview.GetRoot();
root.SetCaption( "new caption" );
```

The following VB.NET sample changes the caption for the root node:

```
With AxChartView1.Root
   .Caption = "new caption"
End With
```

The following C# sample changes the caption for the root node:

```
EXORGCHARTLib.Node root = axChartView1.Root;
root.Caption = "new caption";
```

The following VFP sample changes the caption for the root node:

```
with thisform.ChartView1.Root
   .Caption = "new caption"
endwith
```

property Node.CaptionSingleLine as CaptionSingleLineEnum

Specifies if the element's caption is displayed on single or multiple lines.

Type	Description
CaptionSingleLineEnum	A CaptionSingleLineEnum expression that specifies whether the node's caption is displayed on single or multiple lines.

By default, the CaptionSingleLine property is exCaptionWordWrap. The CaptionSingleLine property specifies whether the node's caption is displayed on single or multiple lines. Use the Caption property to define the label or caption to be displayed on the element's background. The Images method loads icons to be displayed on the control's surface. The HTMLPicture property loads and assigns a picture to a key to be used on control's surface.

method Node.ClearAssistants ()

Clears the assistant nodes.

Type	Description

The ClearAssistants method clears the assistant nodes collection. Use the AddAssistant method to add an assistant node. Use the RemoveAssistant method to remove a specific assistant node. The IsAssistant property determines whether a node is an assistant node or a child node. Use the Clear method to clear the child nodes. Use the ShowAssistants property to hide all assistant nodes.

method Node.ClearBackColor ()

Clears the node's background color.

Type	Description

Use the ClearBackColor property to clear the node's background color previously defined by the BackColor property. If the node's background color is cleared using the ClearBackColor method the node's background color is defined by the BackColorNode property. Use the BackColor property to specify the control's background.

The following VB sample changes the background and foreground color for the selected node:

```
Private Sub ChartView1_Select(ByVal OldNode As EXORGCHARTLibCtl.INode, ByVal
NewNode As EXORGCHARTLibCtl.INode)
   If Not (OldNode Is Nothing) Then
      With OldNode
         .ClearBackColor
         .ClearForeColor
      End With
   End If
   With NewNode
      .ForeColor = vbWhite
      .BackColor = vbBlue
   End With
End Sub
```

The following C++ sample changes the background and foreground color for the selected node:

```
void OnSelectChartview1(LPDISPATCH OldNode, LPDISPATCH NewNode)
{
   CNode oldNode( OldNode ); oldNode.m_bAutoRelease = FALSE;
   CNode newNode( NewNode ); newNode.m_bAutoRelease = FALSE;

   if ( oldNode.m_lpDispatch != NULL )
   {
      oldNode.ClearBackColor();
      oldNode.ClearForeColor();
```

```
    }
    newNode.SetBackColor( RGB(0,0,128) );
    newNode.SetForeColor( RGB(255,255,255) );
}
```

The following VB.NET sample changes the background and foreground color for the selected node:

```
Private Sub AxChartView1_SelectEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_SelectEvent) Handles AxChartView1.SelectEvent
    If Not (e.oldNode Is Nothing) Then
        With e.oldNode
            .ClearBackColor()
            .ClearForeColor()
        End With
    End If
    With e.newNode
        .ForeColor = ToUInt32(Color.White)
        .BackColor = ToUInt32(Color.Blue)
    End With
End Sub
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background and foreground color for the selected node:

```
private void axChartView1_SelectEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_SelectEvent e)
{
    if ( e.oldNode != null )
```

```
    {
        e.oldNode.ClearBackColor();
        e.oldNode.ClearForeColor();
    }
    e.newNode.BackColor = ToUInt32(Color.Blue);
    e.newNode.ForeColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background and foreground color for the selected node:

```
*** ActiveX Control Event ***
LPARAMETERS oldnode, newnode

If !isnull(oldnode)
    With oldnode
        .ClearBackColor
        .ClearForeColor
    EndWith
EndIf
With newnode
    .ForeColor = RGB(255,255,255)
    .BackColor = RGB(0,0,128)
EndWith
```

method Node.ClearForeColor ()

Clears the node's foreground color.

Type	Description

Use the ClearForeColor property to clear the node's foreground color previously defined by the ForeColor property. If the node's foreground color is cleared using the ClearForeColor method the node's foreground color is defined by the ForeColorNode property. Use the ForeColor property to specify the control's foreground.

The following VB sample changes the background and foreground color for the selected node:

```
Private Sub ChartView1_Select(ByVal OldNode As EXORGCHARTLibCtl.INode, ByVal
NewNode As EXORGCHARTLibCtl.INode)
   If Not (OldNode Is Nothing) Then
      With OldNode
         .ClearBackColor
         .ClearForeColor
      End With
   End If
   With NewNode
      .ForeColor = vbWhite
      .BackColor = vbBlue
   End With
End Sub
```

The following C++ sample changes the background and foreground color for the selected node:

```
void OnSelectChartview1(LPDISPATCH OldNode, LPDISPATCH NewNode)
{
   CNode oldNode( OldNode ); oldNode.m_bAutoRelease = FALSE;
   CNode newNode( NewNode ); newNode.m_bAutoRelease = FALSE;

   if ( oldNode.m_lpDispatch != NULL )
   {
      oldNode.ClearBackColor();
      oldNode.ClearForeColor();
```

```
    }
    newNode.SetBackColor( RGB(0,0,128) );
    newNode.SetForeColor( RGB(255,255,255) );
}
```

The following VB.NET sample changes the background and foreground color for the selected node:

```
Private Sub AxChartView1_SelectEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_SelectEvent) Handles AxChartView1.SelectEvent
    If Not (e.oldNode Is Nothing) Then
        With e.oldNode
            .ClearBackColor()
            .ClearForeColor()
        End With
    End If
    With e.newNode
        .ForeColor = ToUInt32(Color.White)
        .BackColor = ToUInt32(Color.Blue)
    End With
End Sub
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background and foreground color for the selected node:

```
private void axChartView1_SelectEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_SelectEvent e)
{
    if ( e.oldNode != null )
```

```
    {
        e.oldNode.ClearBackColor();
        e.oldNode.ClearForeColor();
    }
    e.newNode.BackColor = ToUInt32(Color.Blue);
    e.newNode.ForeColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background and foreground color for the selected node:

```
*** ActiveX Control Event ***
LPARAMETERS oldnode, newnode

If !isnull(oldnode)
    With oldnode
        .ClearBackColor
        .ClearForeColor
    EndWith
EndIf
With newnode
    .ForeColor = RGB(255,255,255)
    .BackColor = RGB(0,0,128)
EndWith
```

method Node.ClearGroup ()

Clears the nodes in the same group.

Type	Description

property Node.CountAssistants as Long

Specifies the number of assistant nodes.

Type	Description
Long	A long expression that indicates the number of assistant nodes.

The CountAssistants property counts the assistant nodes assigned to the node. Use the AddAssistant method to add an assistant node. Use the RemoveAssistant node to remove an assistant node. Use the Assistant property to get an assistant node based on its index. Use the IsAssistant property to specify whether the node is an assistant node or a child node. Use the ShowAssistants property to hide all assistant nodes.

The following VB sample enumerates the assistant nodes, for the root node:

```
With ChartView1.Root
    For i = 0 To .CountAssistants - 1
      Debug.Print .Assistant(i).Caption
    Next
End With
```

The following C++ sample enumerates the assistant nodes, for the root node:

```
CNode root = m_chartview.GetRoot();
for ( long i = 0; i < root.GetCountAssistants(); i++ )
{
  CNode assistant = root.GetAssistant( COleVariant( i ) );
  OutputDebugString( assistant.GetCaption() );
}
```

The following VB.NET sample enumerates the assistant nodes, for the root node:

```
With AxChartView1.Root
  Dim i As Integer
  For i = 0 To .CountAssistants - 1
    Debug.WriteLine(.Assistant(i).Caption())
  Next
End With
```

The following C# sample enumerates the assistant nodes, for the root node:

```
EXORGCHARTLib.Node root = axChartView1.Root;
for (int i = 0; i < root.CountAssistants; i++)
{
   EXORGCHARTLib.Node node = root.get_Assistant(i);
   System.Diagnostics.Debug.WriteLine(node.Caption);
}
```

The following VFP sample enumerates the assistant nodes, for the root node:

```
with thisform.ChartView1.Root
   For i = 0 To .CountAssistants - 1
      wait window  nowait .Assistant(i).Caption
   Next
endwith
```

property Node.CountGroup as Long

Specifies the number of nodes in the same group.

Type	Description
Long	A long expression that specifies the number of nodes in the same group.

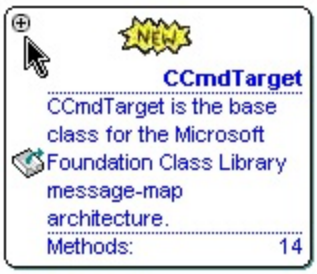
Use the CountGroup and Group properties to access or enumerate the group nodes collection. Use the AddGroup method to add new nodes in the same group (multiple parents). Use the InflateGroupY property to specify the indentation of the group on vertical axis. Use the IsGroup property to specify whether a node was added using the AddGroup method.

property Node.DrawRoundNode as Boolean

Specifies a value that indicates whether the node has borders with round corners.

Type	Description
Boolean	A boolean expression that indicates whether the node has borders with round corners.

Use the DrawRoundNode property to specify whether the node has borders with round corners. Use the DrawRoundNode property to specify define round corners for all nodes in the organigram. The ShadowNode property determines whether the control displays a shadow for nodes. Use the ShadowNode property to hide the shadow for a specific node. Use the SelColor property to specify the color to mark the selected node. Use the SelectNode property to specify the selected node. Use the BackColor property to specify the control's background color. Use the PenBorderNode property to define the type of the pen used to paint the borders for a specified node. Use the BorderColor property to assign a different color for the node's border.



property Node.Editable as EditableNodeEnum

Specifies whether the node's caption is editable.

Type	Description
EditableNodeEnum	An EditableNodeEnum expression that specifies if the node is editable at runtime.

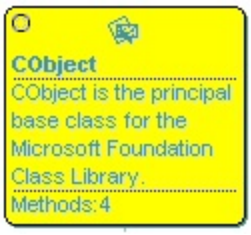
By default, the Editable property is exEditable. Use the Editable property to specify whether a specified node can be edited at runtime. The Caption property indicates the caption of the node being edited. The AllowEdit property specifies the combination of keys that allows the user to edit a node. The LayoutStartChanging(exEditNode) event notifies your application once the user starts editing the node's caption. The LayoutEndChanging(exEditNode) event notifies your application once of the edit operation ends. The Background(exEditNodeBackColor)/Background(exEditNodeForeColor) property specifies the background/foreground color of the edit field being displayed on the node while editing. Use the EditNode to programmatically edit the giving node.

property Node.Enabled as Boolean

Enables or disables the node.

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

By default, the node is enabled. Use the Enabled property to disable the node. If a node is disabled, it is not selectable. A disabled node shows grayed the node's caption and the node's icon. Use the Caption property to specify the caption of the node. Use the Image property to assign an icon to a node. Use the Picture property to assign a custom size picture to a node. Use the Enabled property to disable the control. Use the BackColor property to specify the node's background color.

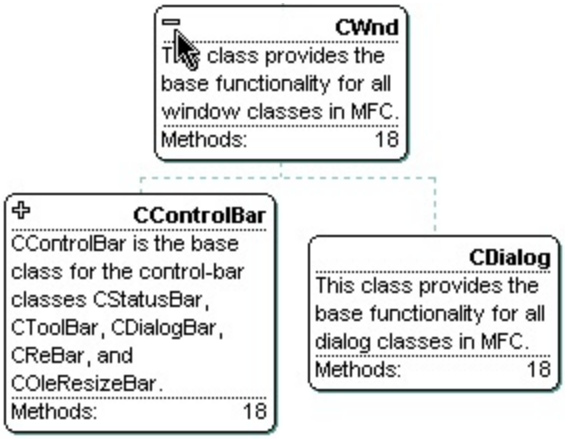


property Node.Expanded as Boolean

Sets or returns whether a node in the hierarchy is expanded.

Type	Description
Boolean	A boolean expression that indicates whether a node is expanded or collapsed.

By default, all nodes are expanded. The control fires the Expand event when the user expands or collapses a node. Use the Expanded property to expand nodes by code. Use the HasButtons property to display the +/- buttons for parent nodes. Use the ExpandOnDblClick property to specify whether a node is expanded or collapsed when the user double clicks it. Use the EnsureVisibleNode method to ensure that a node fits the control's visible client area. Use the SelectNode property to select a node.



property Node.FirstNode as Node

Gets the first child node in the node collection.

Type	Description
Node	A Node object that identifies the first child node.

Use the FirstNode property to get the first child node. The FirstNode property gets nothing, if the node has no child nodes. Use the NextNode property to determine the next sibling node. Use the NodeCount property to get the number of child nodes. Use the Position property to change the node's position. Use the LastNode property to determine the last child node. Use the Nodes property to access the nodes collection. Use the Add method to add a child node.

The following VB sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLibCtl.Node)
   Dim c As EXORGCHARTLibCtl.Node
   Set c = n.FirstNode
   While Not c Is Nothing
      Debug.Print c.Caption
      enumRec c
      Set c = c.NextNode
   Wend
End Sub
```

The following C++ sample enumerates recursively all the child nodes:

```
void enumRec( CNode* pNode )
{
   if ( pNode != NULL )
   {
      CNode child = pNode->GetFirstNode();
      while ( child.m_lpDispatch != NULL )
      {
         OutputDebugString( child.GetCaption() );
         OutputDebugString( "\r\n" );
         enumRec( &child );
         child = child.GetNextNode();
      }
```

```
    }
}
```

The following VB.NET sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLib.Node)
    Dim c As EXORGCHARTLib.Node = n.FirstNode
    While Not c Is Nothing
        Debug.Print(c.Caption)
        enumRec(c)
        c = c.NextNode
    End While
End Sub
```

The following C# sample enumerates recursively all the child nodes:

```
private void enumRec(EXORGCHARTLib.Node node)
{
    EXORGCHARTLib.Node child = node.FirstNode;
    while (child != null)
    {
        System.Diagnostics.Debug.WriteLine(child.Caption);
        enumRec(child);
        child = child.NextNode;
    }
}
```

The following VFP sample enumerates recursively all the child nodes:

```
LPARAMETERS node

local child
child = node.FirstNode
do while ( !isnull( child ) )
    wait window nowait child.Caption
    thisform.enumrec( child )
    child = child.NextNode
enddo
```

property Node.FixedHeight as Long

Retrieves or sets a value that indicates whether the height of the node's caption is fixed.

Type	Description
Long	A long expression that defines the fixed width for the node's caption.

By default, the FixedHeight property is -1. Use the FixedWidth and FixedHeight property to define the fixed size for the caption of the node. If the FixedHeight property is negative, the control gets automatically the height of the node, to let the entire caption being visible. Use the Font property to specify the control's font. Use the FixedWidthHeight and FixedHeightHeight properties to specify fixed size for all nodes in the organigram. Use the Padding property of the Node to define the padding for specified node.

property Node.FixedWidth as Long

Retrieves or sets a value that indicates whether the width of the node's caption is fixed.

Type	Description
Long	A long expression that defines the fixed width of the caption of the node.

Use the FixedWidth and FixedHeight property to define the fixed size for the caption of the node. The FixedWidth property is -1. If the FixedWidth property is negative, the control gets automatically the width of the node, to let the entire caption being visible. In this case, the Width property defines the maximum width for the node's caption. Use the Font property to specify the control's font. Use the FixedWidthHeight and FixedHeightHeight properties to specify fixed size for all nodes in the organigram. Use the Caption property to specify the node's caption. Use the Padding property of the Node to define the padding for specified node.

property Node.ForeColor as Color

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

Type	Description
Color	A color expression that defines the node's foreground color.

Use the ForeColor property to specify the node's foreground color. Use the ForeColorNode property to define the default foreground color for nodes. Use the ClearForeColor method to clear the node's foreground color. Use the ForeColor property to specify the control's foreground.

The following VB sample changes the background and foreground color for the selected node:

```
Private Sub ChartView1_Select(ByVal OldNode As EXORGCHARTLibCtl.INode, ByVal
NewNode As EXORGCHARTLibCtl.INode)
   If Not (OldNode Is Nothing) Then
      With OldNode
         .ClearBackColor
         .ClearForeColor
      End With
   End If
   With NewNode
      .ForeColor = vbWhite
      .BackColor = vbBlue
   End With
End Sub
```

The following C++ sample changes the background and foreground color for the selected node:

```
void OnSelectChartview1(LPDISPATCH OldNode, LPDISPATCH NewNode)
{
   CNode oldNode( OldNode ); oldNode.m_bAutoRelease = FALSE;
   CNode newNode( NewNode ); newNode.m_bAutoRelease = FALSE;

   if ( oldNode.m_lpDispatch != NULL )
   {
```

```
        oldNode.ClearBackColor();
        oldNode.ClearForeColor();
    }
    newNode.SetBackColor( RGB(0,0,128) );
    newNode.SetForeColor( RGB(255,255,255) );
}
```

The following VB.NET sample changes the background and foreground color for the selected node:

```
Private Sub AxChartView1_SelectEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_SelectEvent) Handles AxChartView1.SelectEvent
    If Not (e.oldNode Is Nothing) Then
        With e.oldNode
            .ClearBackColor()
            .ClearForeColor()
        End With
    End If
    With e.newNode
        .ForeColor = ToUInt32(Color.White)
        .BackColor = ToUInt32(Color.Blue)
    End With
End Sub
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background and foreground color for the selected node:

```
private void axChartView1_SelectEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_SelectEvent e)
```

```
{
   if ( e.oldNode != null )
   {
      e.oldNode.ClearBackColor();
      e.oldNode.ClearForeColor();
   }
   e.newNode.BackColor = ToUInt32(Color.Blue);
   e.newNode.ForeColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background and foreground color for the selected node:

```
*** ActiveX Control Event ***
LPARAMETERS oldnode, newnode

If !isnull(oldnode)
   With oldnode
      .ClearBackColor
      .ClearForeColor
   EndWith
EndIf
With newnode
   .ForeColor = RGB(255,255,255)
   .BackColor = RGB(0,0,128)
EndWith
```

property Node.Group (Index as Variant) as Node

Retrieves an node in the group by its index.

Type	Description
Index as Variant	A long expression that specifies the index of the node in the group being requested.
Node	A Node object being requested.

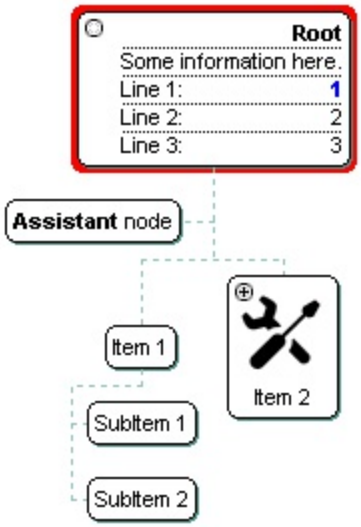
Use the CountGroup and Group properties to access or enumerate the group nodes collection. Use the AddGroup method to add new nodes in the same group (multiple parents). Use the InflateGroupY property to specify the indentation of the group on vertical axis. Use the IsGroup property to specify whether a node was added using the AddGroup method.

property Node.HasButton as Boolean

Specifies whether the node displays the +/- buttons.

Type	Description
Boolean	A boolean expression that indicates whether the node displays the +/- button.

By default, the HasButton property is True. Use the HasButton property to hide the +/- buttons for a particular node. Use the HasButtons property to display +/- buttons for parent nodes. The node displays the +/- button only if the node contains child nodes, and the HasButton property is True. Use the ExpandOnDblClick property to specify whether a node is expanded or collapsed when the user double clicks it. Use the EnsureVisibleNode method to ensure that a node fits the control's visible client area. Use the SelectNode property to select a node. The Expanded property sets or returns whether a node in the hierarchy is expanded. Use the Parent property to get the parent node. Use the ButtonsAlign property to align the +/- expand buttons in the node.



property Node.Image as Long

Specifies a value that indicates the index of image being used.

Type	Description
Long	A long expression that identifies the index of image in the Images collection that's displayed in the node.

Use the Image property to assign an 16x16 icon to the node. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the Picture property to associate a picture to a node. Use the Images property to access the control's Images collection. You can assign the Image property when the Add method is called. Use the ImageAlignment property to align the node's icon. Use the Images method to assign a list of icons at runtime. Use the ReplaceIcon property to add new icons to the control's images list. Use the **<img>** HTML tag to insert icons inside the node's caption.

The following VB sample assigns an icon to the root node:

```
With ChartView1.Root
    .Caption = "<r><dotline><b><fgcolor=0000FF>Andrew Fuller</fgcolor></b>
<br><solidline><b>Title</b>:<r><fgcolor=FF0000>Vice President Sales</fgcolor>
<br>USA, Tacoma, WA, 98401, 908 W. Capital Way<br><dotline><upline><b>Phone:
</b><r>(206) 555-9482"
    .Image = 1
    .ImageAlignment = exImageTop
End With
```

The following C++ sample assigns an icon to the root node:

```
m_chartview.GetRoot().SetImage( 1 );
```

The following VB.NET sample assigns an icon to the root node:

```
With AxChartView1.Root
  .Image = 1
End With
```

The following C# sample assigns an icon to the root node:

```
axChartView1.Root.Image = 1;
```

The following VFP sample assigns an icon to the root node:

```
With thisform.ChartView1.Root
   .Image = 1
EndWith
```

property Node.ImageAlignment as ImageAlignmentEnum

Specifies the alignment of the image within the node.

Type	Description
ImageAlignmentEnum	An ImageAlignmentEnum expression that indicates the alignment of the icon in the node.

Use the Image property to assign an 16x16 icon to the node. Use the Images property to access the control's Images collection. You can assign the Image property when the Add method is called. The ImageAlignment has no effect if the node has no icon associated. Use the Picture property to associate a picture to a node. Use the Images method to assign a list of icons at runtime. Use the ReplaceIcon property to add new icons to the control's images list.

property Node.Index as Long

Gets the index of the node within the control Nodes collection.

Type	Description
Long	A long expression that indicates the index of the node.

The Index property retrieves the node's index. The index of the node is allocated by the control when adding to the Nodes collection using the Add method. A node can be identified by its index or by its key. Use the Count property to indicate the number of nodes in the collection. Use the Item property to access a node by its index or by its key. Use the Root property to access the root node. Use the Key property to access the node's key. The Position property specifies the position of the node. Use the FirstNode property to get the first child node. Use the NextNode property to determine the next sibling node. Use the NodeCount property to get the number of child nodes.

property Node.InflateGroupX as Long

Increases the width of the group.

Type	Description
Long	A long expression that specifies the number of pixels to increase the width of the group.

Use the InflateGroupX property to increase the width of the group. Use the AddGroup method to add new nodes in the same group (multiple parents). Use the CountGroup and Group properties to access or enumerate the group nodes collection. This property has effect if the control's Layout property is exLayoutLTR.

property Node.InflateGroupY as Long

Increases the height of the group.

Type	Description
Long	A long expression that specifies the number of pixels to increase the height of the group.

Use the InflateGroupY property to increase the height of the group. Use the AddGroup method to add new nodes in the same group (multiple parents). Use the CountGroup and Group properties to access or enumerate the group nodes collection. This property has effect if the control's Layout property is exLayoutTTB.

property Node.IsAssistant as Boolean

Retrieves a value that specifies whether the node is an assistant.

Type	Description
Boolean	A boolean expression that indicates whether the node is an assistant node or a child node.

The IsAssistant property gets True, if the node was added using the AddAssistant method, else it gets False. Use the IsAssistant property to determine whether the node is an assistant node or a child node. Use the RemoveAssistant method to remove an assistant node. Use the NodeFromPoint property to retrieve the node from the cursor.

The following VB sample prints the caption of the assistant node from the cursor:

```
Private Sub ChartView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .NodeFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (n Is Nothing) Then
         If (n.IsAssistant) Then
            Debug.Print n.Caption
         End If
      End If
   End With
End Sub
```

The following C++ sample prints the caption of the assistant node from the cursor:

```
void OnMouseMoveChartview1(short Button, short Shift, long X, long Y)
{
   CNode node = m_chartview.GetNodeFromPoint( X, Y );
   if ( node.m_lpDispatch != NULL )
      if ( node.GetIsAssistant() )
         OutputDebugString( node.GetCaption() );
}
```

The following VB.NET sample prints the caption of the assistant node from the cursor:

```
Private Sub AxChartView1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent) Handles
AxChartView1.MouseMoveEvent
   With AxChartView1
      Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(e.x, e.y)
      If Not (n Is Nothing) Then
         If (n.IsAssistant) Then
            Debug.WriteLine(n.Caption)
         End If
      End If
   End With
End Sub
```

The following C# sample prints the caption of the assistant node from the cursor:

```
private void axChartView1_MouseMoveEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent e)
{
   EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(e.x, e.y);
   if (node != null)
      if ( node.IsAssistant )
         System.Diagnostics.Debug.WriteLine(node.Caption);
}
```

The following VFP sample prints the caption of the assistant node from the cursor:

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

With thisform.ChartView1
   local n
   n = .NodeFromPoint(x , y )
   If !isnull(n) then
      if ( n.IsAssistant )
         wait window nowait n.Caption
      Endif
   EndIf
EndWith
```

property Node.IsGroup as Boolean

Retrieves a value that specifies whether the node belongs to a group.

Type	Description
Boolean	A Boolean expression that specifies whether the node is added using the AddGroup method or not.

Use the IsGroup property to specify whether a node was added using the AddGroup method. Use the CountGroup and Group properties to access or enumerate the group nodes collection. Use the AddGroup method to add new nodes in the same group (multiple parents). Use the InflateGroupY property to specify the indentation of the group on vertical axis.

property Node.Key as String

Specifies the node's key.

Type	Description
String	A string expression that identifies the key of the node.

A node can be identified by its index or by its key. Use the Add method to assign a key to a node. Adding two nodes with the same key fails. Use the Item property to access an item by its key. Use the Root property to get the root node. Use the Key property to assign a new key to your root node like in the following VB sample:

```
With ChartView1.Root
   .Key = "newKeyForRoot"
End With
```

Assigning a new key for a node fails if the new key is already assigned to another node, or if it is empty.

The following C++ sample assigns a new key for the root node:

```
m_chartview.GetRoot().SetKey("newrootkey");
```

The following VB.NET sample assigns a new key for the root node:

```
With AxChartView1
   .Root.Key = "newrootkey"
End With
```

The following C# sample assigns a new key for the root node:

```
axChartView1.Root.Key = "newrootkey";
```

The following VFP sample assigns a new key for the root node:

```
With thisform.ChartView1
   .Root.Key = "newrootkey"
EndWith
```

property Node.LastNode as Node

Gets the last child node.

Type	Description
Node	A Node position that specifies the last child node.

The LastNode property determines the last child node. If the node has no child nodes, the LastNode property gets nothing. Use the FirstNode property to determine the first child node. Use the NodeCount property to get the number of child nodes. Use the Position property to change the node's position. Use the Nodes property to access the nodes collection. Use the Add method to add a child node.

The following VB sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLibCtl.Node)
   Dim c As EXORGCHARTLibCtl.Node
   Set c = n.FirstNode
   While Not c Is Nothing
      Debug.Print c.Caption
      enumRec c
      Set c = c.NextNode
   Wend
End Sub
```

The following C++ sample enumerates recursively all the child nodes:

```
void enumRec( CNode* pNode )
{
   if ( pNode != NULL )
   {
      CNode child = pNode->GetFirstNode();
      while ( child.m_lpDispatch != NULL )
      {
         OutputDebugString( child.GetCaption() );
         OutputDebugString( "\r\n" );
         enumRec( &child );
         child = child.GetNextNode();
      }
   }
```

```
}
```

The following VB.NET sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLib.Node)
   Dim c As EXORGCHARTLib.Node = n.FirstNode
   While Not c Is Nothing
      Debug.Print(c.Caption)
      enumRec(c)
      c = c.NextNode
   End While
End Sub
```

The following C# sample enumerates recursively all the child nodes:

```
private void enumRec(EXORGCHARTLib.Node node)
{
   EXORGCHARTLib.Node child = node.FirstNode;
   while (child != null)
   {
      System.Diagnostics.Debug.WriteLine(child.Caption);
      enumRec(child);
      child = child.NextNode;
   }
}
```

The following VFP sample enumerates recursively all the child nodes:

```
LPARAMETERS node

local child
child = node.FirstNode
do while ( !isnull( child ) )
   wait window nowait child.Caption
   thisform.enumrec( child )
   child = child.NextNode
enddo
```

property Node.Left as Boolean

Retrieves or sets a value that indicates whether the assistant node is on the left side.

Type	Description
Boolean	A boolean expression that indicates whether the assistant node is aligned to the left side.

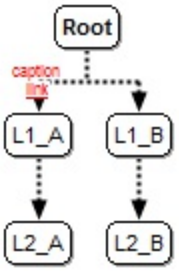
The Left property has effect only for assistant nodes. Use the AddAssistant method to add assistant nodes. Use the IsAssistant property to determine whether a node is an assistant node or a child node. Use the Assistant property to access the assistant nodes collection. Use the RemoveAssistant method to remove an assistant node. Use the Caption property to change the node's caption at runtime. *Please notice that the assistant node is always aligned to the right, no matter of the Left property, if is a child of a node that has the ArrangeSiblingNodesAs property on exTree.*

property Node.LinkCaption as String

Specifies the caption on the node's link.

Type	Description
String	A string expression that indicates the caption on the link between the parent and the current node.

Use the LinkCaption to specify a HTML caption, icon, pictures or anchors on the regular links. The regular link is shown between the parent and the current node. Use the Caption property to specify the node's caption. Use the Image property to assign an icon to a node. Use the Picture property to assign a custom size picture to a node. You can use the LinkCaptionFrompoint property to get the node whose caption on the link is at specified position.



The LinkCaption property supports built-in HTML format. The supported tags are:

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

  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=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part

of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

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

gradient-center

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

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the

color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

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

outline anti-aliasing

The following samples shows how you can assign a HTML caption to a link:

VBA (MS Access, Excell...)

```
With ChartView1
   .IndentSiblingY = 30
   .ShowLinksDir = True
   .PenWidthLink = 2
   .LinkColor = RGB(0,0,0)
   .AntiAliasing = True
   With .Nodes
      .Add("L1_A",0,"LA").LinkCaption = "<fgcolor=FF0000><font ;6>caption<br>
<c><bgcolor=FFFFFF><a>link"
      .Add "L1_B",0,"LB"
      .Add "L2_A","LA","LA2"
      .Add "L2_B","LB","LB2"
   End With
End With
```

VB6

```
With ChartView1
   .IndentSiblingY = 30
   .ShowLinksDir = True
   .PenWidthLink = 2
   .LinkColor = RGB(0,0,0)
   .AntiAliasing = True
   With .Nodes
```

```
        .Add("L1_A",0,"LA").LinkCaption = "<fgcolor=FF0000><font ;6>caption<br>
<c><bgcolor=FFFFFF><a>link"
        .Add "L1_B",0,"LB"
        .Add "L2_A","LA","LA2"
        .Add "L2_B","LB","LB2"
    End With
End With
```

VB.NET

```
With Exchartview1
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = Color.FromArgb(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",0,"LA").LinkCaption = "<fgcolor=FF0000><font ;6>caption<br>
<c><bgcolor=FFFFFF><a>link"
        .Add("L1_B",0,"LB")
        .Add("L2_A","LA","LA2")
        .Add("L2_B","LB","LB2")
    End With
End With
```

VB.NET for /COM

```
With AxChartView1
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",0,"LA").LinkCaption = "<fgcolor=FF0000><font ;6>caption<br>
<c><bgcolor=FFFFFF><a>link"
        .Add("L1_B",0,"LB")
        .Add("L2_A","LA","LA2")
```

```
    .Add("L2_B","LB","LB2")
  End With
End With
```

C++

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

  #import <ExOrgChart.dll>
  using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->PutIndentSiblingY(30);
spChartView1->PutShowLinksDir(VARIANT_TRUE);
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(RGB(0,0,0));
spChartView1->PutAntiAliasing(VARIANT_TRUE);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
  var_Nodes->Add(L"L1_A",long(0),"LA",vtMissing,vtMissing)->PutLinkCaption(L"
<fgcolor=FF0000><font ;6>caption<br><c><bgcolor=FFFFFF><a>link");
  var_Nodes->Add(L"L1_B",long(0),"LB",vtMissing,vtMissing);
  var_Nodes->Add(L"L2_A","LA","LA2",vtMissing,vtMissing);
  var_Nodes->Add(L"L2_B","LB","LB2",vtMissing,vtMissing);
```

C#

```
exchartview1.IndentSiblingY = 30;
exchartview1.ShowLinksDir = true;
exchartview1.PenWidthLink = 2;
exchartview1.LinkColor = Color.FromArgb(0,0,0);
exchartview1.AntiAliasing = true;
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
  var_Nodes.Add("L1_A",0,"LA",null,null).LinkCaption = "<fgcolor=FF0000><font
```

```
;6>caption<br><c><bgcolor=FFFFFF><a>link";
   var_Nodes.Add("L1_B",0,"LB",null,null);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
   var_Nodes.Add("L2_B","LB","LB2",null,null);
```

C# for /COM

```
axChartView1.IndentSiblingY = 30;
axChartView1.ShowLinksDir = true;
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
axChartView1.AntiAliasing = true;
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   var_Nodes.Add("L1_A",0,"LA",null,null).LinkCaption = "<fgcolor=FF0000><font
;6>caption<br><c><bgcolor=FFFFFF><a>link";
   var_Nodes.Add("L1_B",0,"LB",null,null);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
   var_Nodes.Add("L2_B","LB","LB2",null,null);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Node,com_Nodes;
   anytype var_Node,var_Nodes;
   ;

   super();

   exchartview1.IndentSiblingY(30);
   exchartview1.ShowLinksDir(true);
   exchartview1.PenWidthLink(2);
   exchartview1.LinkColor(WinApi::RGB2int(0,0,0));
   exchartview1.AntiAliasing(true);
   var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
      var_Node =
```

```
COM::createFromObject(com_Nodes.Add("L1_A",COMVariant::createFromInt(0),"LA"));
 com_Node = var_Node;
      com_Node.LinkCaption("<fgcolor=FF0000><font ;6>caption<br><c>
<bgcolor=FFFFFF><a>link");
      com_Nodes.Add("L1_B",COMVariant::createFromInt(0),"LB");
      com_Nodes.Add("L2_A","LA","LA2");
      com_Nodes.Add("L2_B","LB","LB2");
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
   IndentSiblingY := 30;
   ShowLinksDir := True;
   PenWidthLink := 2;
   LinkColor := Color.FromArgb(0,0,0);
   AntiAliasing := True;
   with Nodes do
   begin
      Add('L1_A',TObject(0),'LA',Nil,Nil).LinkCaption := '<fgcolor=FF0000><font
;6>caption<br><c><bgcolor=FFFFFF><a>link';
      Add('L1_B',TObject(0),'LB',Nil,Nil);
      Add('L2_A','LA','LA2',Nil,Nil);
      Add('L2_B','LB','LB2',Nil,Nil);
   end;
end
```

Delphi (standard)

```
with ChartView1 do
begin
   IndentSiblingY := 30;
   ShowLinksDir := True;
   PenWidthLink := 2;
   LinkColor := RGB(0,0,0);
   AntiAliasing := True;
   with Nodes do
```

```
  begin
    Add('L1_A',OleVariant(0),'LA',Null,Null).LinkCaption := '<fgcolor=FF0000><font
;6>caption<br><c><bgcolor=FFFFFF><a>link';
    Add('L1_B',OleVariant(0),'LB',Null,Null);
    Add('L2_A','LA','LA2',Null,Null);
    Add('L2_B','LB','LB2',Null,Null);
  end;
end
```

VFP

```
with thisform.ChartView1
  .IndentSiblingY = 30
  .ShowLinksDir = .T.
  .PenWidthLink = 2
  .LinkColor = RGB(0,0,0)
  .AntiAliasing = .T.
  with .Nodes
    .Add("L1_A",0,"LA").LinkCaption = "<fgcolor=FF0000><font ;6>caption<br>
<c><bgcolor=FFFFFF><a>link"
    .Add("L1_B",0,"LB")
    .Add("L2_A","LA","LA2")
    .Add("L2_B","LB","LB2")
  endwith
endwith
```

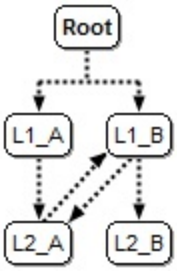
property Node.LinkTo as Variant

Retrieves or sets a value that indicates the list of nodes that the source node links to.

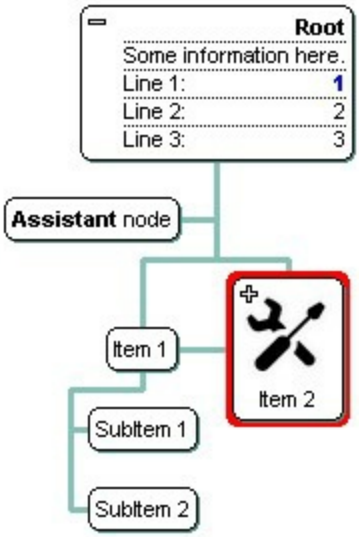
Type	Description
Variant	A string expression that indicates the list of nodes that the current node links to. The list of nodes (keys) is separated by ',' character

Use the LinkTo property to connect a node with multiple nodes. By default, the LinkTo property is empty. The LinkTo property indicates a list of keys separated by comma, that links to the current node. For instance, if the LinkTo property is "Key1,Key2", the current node draws a link to the nodes with the following keys: Key1 and Key2. Use the Key property to assign a key to a node. The PenWidthLinkTo property specifies the thickness of the lines between nodes. Use the PenLinkTo property to specify the type of the pen used to paint the lines between nodes. The LinkToColor property specifies the color of the links between nodes. The ShowLinks property doesn't affect the links added with the LinkTo property. The LinkToCaption property specifies the HTML caption being shown on the links between nodes. The ShowLinksDir property specifies whether the links between nodes show their directions.

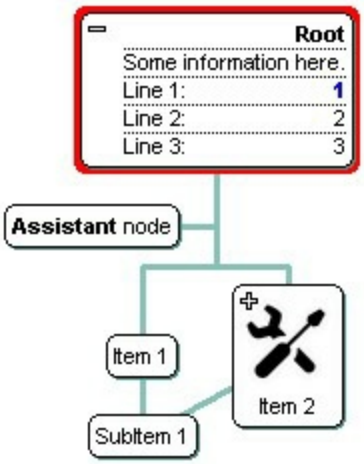
The following screen shot shows a bi-directional linkto as L1_B node links to L2_A and back:



In the following screen shot you can notice that the "Item 1" node is linked directly to the "Item 2" node.



In the following screen shot you can notice that the "Subitem 1" node is linked to the "Item 2" node.



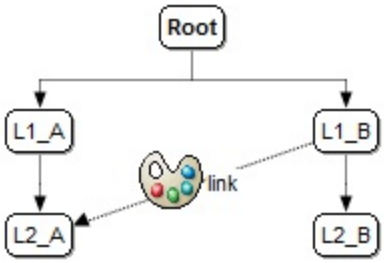
property Node.LinkToCaption(Key as String) as String

Specifies the HTML caption being shown on a LinkTo line.

Type	Description
Key as String	A String expression that specifies the key of the node to link to
String	A String expression that indicates the HTML caption being shown on the link.

Use the LinkToCaption property to specify the caption being shown on the LinkTo property. The LinkTo property adds an arbitrary link between two nodes. The LinkColor property specifies the color of the links between nodes. The ShowLinks property doesn't affect the links added with the LinkTo property. The ShowLinksDir property specifies whether the links between nodes show their directions. The PenWidthLink property specifies the thickness of the lines between nodes. Use the PenLink property to specify the type of the pen used to paint the lines between nodes.

The following screen shot shows the links between nodes with their direction including HTML captions on links:

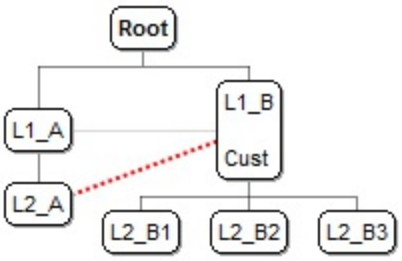


property Node.LinkToColor(Key as String) as Color

Specifies the color to show the LinkTo line.

Type	Description
Key as String	A String expression that specifies the key of the node to link to.
Color	A Color expression that specifies the color to show the link

The LinkToColor property specifies the color for an linkto line. Use the LinkTo property to arbitrary link the current nodes with others. The LinkToColor property specifies the color of the links between nodes. The LinkToWidth, LinkToPen, LinkToRound, LinkToShowDir property specifies the width, pen, roundness, direction to show an arbitrary link.



The following VB sample changes the color to show an arbitrary line::

```
With ChartView1
   .PenWidthLink = 2
   .LinkColor = 0
   With .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      With .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = RGB(255,0,0)
         .LinkToWidth("LA2") = 2
      End With
      .Add "L2_A","LA","LA2"
      .Add "L2_B1","LB","LB21"
      .Add "L2_B2","LB","LB22"
      .Add "L2_B3","LB","LB23"
   End With
End With
```

The following VB.NET sample changes the color to show an arbitrary line::

```
With AxChartView1
   .PenWidthLink = 2
   .LinkColor = Color.FromArgb(0,0,0)
   With .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      With .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = 255
         .LinkToWidth("LA2") = 2
      End With
      .Add "L2_A","LA","LA2"
      .Add "L2_B1","LB","LB21"
      .Add "L2_B2","LB","LB22"
      .Add "L2_B3","LB","LB23"
   End With
End With
```

The following C++ sample changes the color to show an arbitrary line::

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

   #import <ExOrgChart.dll>
   using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(0);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"L1_A",0,"LA",vtMissing,vtMissing)->PutLinkTo("LB");
   EXORGCHARTLib::INodePtr var_Node = var_Nodes->Add(L"L1_B<br>
<br>Cust",0,"LB",vtMissing,vtMissing);
      var_Node->PutLinkTo("LA2");
      var_Node->PutLinkToColor(L"LA2",RGB(255,0,0));
```

```
    var_Node->PutLinkToWidth(L"LA2",2);
  var_Nodes->Add(L"L2_A","LA","LA2",vtMissing,vtMissing);
  var_Nodes->Add(L"L2_B1","LB","LB21",vtMissing,vtMissing);
  var_Nodes->Add(L"L2_B2","LB","LB22",vtMissing,vtMissing);
  var_Nodes->Add(L"L2_B3","LB","LB23",vtMissing,vtMissing);
```

The following C# sample changes the color to show an arbitrary line::

```
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
  var_Nodes.Add("L1_A",0,"LA",null,null).LinkTo = "LB";
  EXORGCHARTLib.Node var_Node = var_Nodes.Add("L1_B<br>
<br>Cust",0,"LB",null,null);
    var_Node.LinkTo = "LA2";
    var_Node.set_LinkToColor("LA2",255);
    var_Node.set_LinkToWidth("LA2",2);
  var_Nodes.Add("L2_A","LA","LA2",null,null);
  var_Nodes.Add("L2_B1","LB","LB21",null,null);
  var_Nodes.Add("L2_B2","LB","LB22",null,null);
  var_Nodes.Add("L2_B3","LB","LB23",null,null);
```

The following VFP sample changes the color to show an arbitrary line::

```
with thisform.ChartView1
  .PenWidthLink = 2
  .LinkColor = 0
  with .Nodes
    .Add("L1_A",0,"LA").LinkTo = "LB"
    with .Add("L1_B<br><br>Cust",0,"LB")
      .LinkTo = "LA2"
      .LinkToColor("LA2") = RGB(255,0,0)
      .LinkToWidth("LA2") = 2
    endwith
    .Add("L2_A","LA","LA2")
    .Add("L2_B1","LB","LB21")
    .Add("L2_B2","LB","LB22")
    .Add("L2_B3","LB","LB23")
```

```
  endwith
endwith
```

The following Delphi sample changes the color to show an arbitrary line::

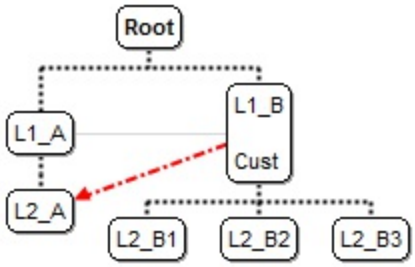
```
with AxChartView1 do
begin
  PenWidthLink := 2;
  LinkColor := Color.FromArgb(0,0,0);
  with Nodes do
  begin
    Add('L1_A',TObject(0),'LA',Nil,Nil).LinkTo := 'LB';
    with Add('L1_B<br><br>Cust',TObject(0),'LB',Nil,Nil) do
    begin
      LinkTo := 'LA2';
      LinkToColor['LA2'] := 255;
      LinkToWidth['LA2'] := 2;
    end;
    Add('L2_A','LA','LA2',Nil,Nil);
    Add('L2_B1','LB','LB21',Nil,Nil);
    Add('L2_B2','LB','LB22',Nil,Nil);
    Add('L2_B3','LB','LB23',Nil,Nil);
  end;
end
```

property Node.LinkToPen(Key as String) as PenTypeEnum

Specifies the style of the link for linkto line.

Type	Description
Key as String	A String expression that specifies the key of the node to link to.
PenTypeEnum	A PenTypeEnum expression that specifies the pen/style to show the link

The LinkToPen property specifies the pen to show an linkto line. Use the LinkTo property to arbitrary link the current nodes with others. The PenLinkTo property specifies the pen of the links between nodes. The LinkToWidth, LinkToColor, LinkToRound, LinkToShowDir property specifies the width, color, roundness, direction to show an arbitrary link.



The following VB sample changes the style for an arbitrary link::

```
With ChartView1
   .ShowLinksDir = False
   .PenWidthLink = 2
   .LinkColor = 0
   .AntiAliasing = True
   With .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      With .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = RGB(255,0,0)
         .LinkToWidth("LA2") = 2
         .LinkToPen("LA2") = exPenDashDot
         .LinkToShowDir("LA2") = True
      End With
      .Add "L2_A","LA","LA2"
      .Add "L2_B1","LB","LB21"
```

```
        .Add "L2_B2","LB","LB22"
        .Add "L2_B3","LB","LB23"
    End With
End With
```

The following VB.NET sample changes the style for an arbitrary link::

```
With AxChartView1
    .ShowLinksDir = False
    .PenWidthLink = 2
    .LinkColor = Color.FromArgb(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",0,"LA").LinkTo = "LB"
        With .Add("L1_B<br><br>Cust",0,"LB")
            .LinkTo = "LA2"
            .LinkToColor("LA2") = 255
            .LinkToWidth("LA2") = 2
            .LinkToPen("LA2") = EXORGCHARTLib.PenTypeEnum.exPenDashDot
            .LinkToShowDir("LA2") = True
        End With
        .Add "L2_A","LA","LA2"
        .Add "L2_B1","LB","LB21"
        .Add "L2_B2","LB","LB22"
        .Add "L2_B3","LB","LB23"
    End With
End With
```

The following C++ sample changes the style for an arbitrary link::

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

    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
```

```
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->PutShowLinksDir(VARIANT_FALSE);
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(0);
spChartView1->PutAntiAliasing(VARIANT_TRUE);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"L1_A",0,"LA",vtMissing,vtMissing)->PutLinkTo("LB");
   EXORGCHARTLib::INodePtr var_Node = var_Nodes->Add(L"L1_B<br>
<br>Cust",0,"LB",vtMissing,vtMissing);
      var_Node->PutLinkTo("LA2");
      var_Node->PutLinkToColor(L"LA2",RGB(255,0,0));
      var_Node->PutLinkToWidth(L"LA2",2);
      var_Node->PutLinkToPen(L"LA2",EXORGCHARTLib::exPenDashDot);
      var_Node->PutLinkToShowDir(L"LA2",VARIANT_TRUE);
   var_Nodes->Add(L"L2_A","LA","LA2",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_B1","LB","LB21",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_B2","LB","LB22",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_B3","LB","LB23",vtMissing,vtMissing);
```

The following C# sample changes the style for an arbitrary link::

```
axChartView1.ShowLinksDir = false;
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
axChartView1.AntiAliasing = true;
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   var_Nodes.Add("L1_A",0,"LA",null,null).LinkTo = "LB";
   EXORGCHARTLib.Node var_Node = var_Nodes.Add("L1_B<br>
<br>Cust",0,"LB",null,null);
      var_Node.LinkTo = "LA2";
      var_Node.set_LinkToColor("LA2",255);
      var_Node.set_LinkToWidth("LA2",2);
      var_Node.set_LinkToPen("LA2",EXORGCHARTLib.PenTypeEnum.exPenDashDot);
      var_Node.set_LinkToShowDir("LA2",true);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
   var_Nodes.Add("L2_B1","LB","LB21",null,null);
```

```
var_Nodes.Add("L2_B2","LB","LB22",null,null);
var_Nodes.Add("L2_B3","LB","LB23",null,null);
```

The following VFP sample changes the style for an arbitrary link::

```
with thisform.ChartView1
   .ShowLinksDir = .F.
   .PenWidthLink = 2
   .LinkColor = 0
   .AntiAliasing = .T.
   with .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      with .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = RGB(255,0,0)
         .LinkToWidth("LA2") = 2
         .LinkToPen("LA2") = 3
         .LinkToShowDir("LA2") = .T.
      endwith
      .Add("L2_A","LA","LA2")
      .Add("L2_B1","LB","LB21")
      .Add("L2_B2","LB","LB22")
      .Add("L2_B3","LB","LB23")
   endwith
endwith
```

The following Delphi sample changes the style for an arbitrary link::

```
with AxChartView1 do
begin
   ShowLinksDir := False;
   PenWidthLink := 2;
   LinkColor := Color.FromArgb(0,0,0);
   AntiAliasing := True;
   with Nodes do
   begin
      Add('L1_A',TObject(0),'LA',Nil,Nil).LinkTo := 'LB';
      with Add('L1_B<br><br>Cust',TObject(0),'LB',Nil,Nil) do
```

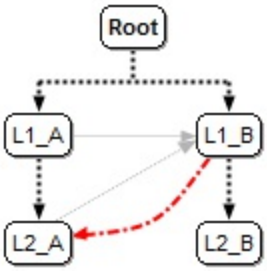
```
      begin
        LinkTo := 'LA2';
        LinkToColor['LA2'] := 255;
        LinkToWidth['LA2'] := 2;
        LinkToPen['LA2'] := EXORGCHARTLib.PenTypeEnum.exPenDashDot;
        LinkToShowDir['LA2'] := True;
      end;
      Add('L2_A','LA','LA2',Nil,Nil);
      Add('L2_B1','LB','LB21',Nil,Nil);
      Add('L2_B2','LB','LB22',Nil,Nil);
      Add('L2_B3','LB','LB23',Nil,Nil);
    end;
end
```

property Node.LinkToRound(Key as String) as Boolean

Specifies whether the LinkTo line is shown linear or round.

Type	Description
Key as String	A String expression that specifies the key of the node to link to.
Boolean	A boolean expression that specifies whether the link is shown linear or round (true).

The LinkToRound property specifies the whether the link is shown liner or round. Use the LinkTo property to arbitrary link the current nodes with others. The ShowRoundLink property specifies whether the links between nodes are shows linear or round. The LinkToWidth, LinkToColor, LinkToShowDir, LinkToPen property specifies the width, color, direction, pen to show an arbitrary link.

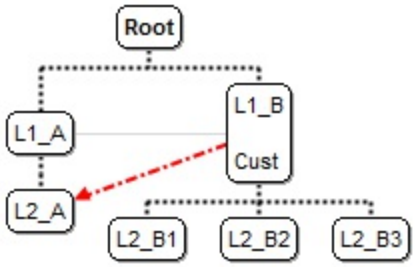


property Node.LinkToShowDir(Key as String) as Boolean

Specifies whether the LinkTo line shows the direction.

Type	Description
Key as String	A String expression that specifies the key of the node to link to.
Boolean	A boolean expression that specifies whether the link shows its direction.

The LinkToShowDir property specifies the whether the link shows its direction . Use the LinkTo property to arbitrary link the current nodes with others. The ShowLinksDir property specifies whether the links between nodes shows the direction. The LinkToWidth, LinkToColor, LinkToRound, LinkToPen property specifies the width, color, roundness, pen to show an arbitrary link.



The following VB sample shows the direction for an arbitrary link only:

```
With ChartView1
   .ShowLinksDir = False
   .PenWidthLink = 2
   .LinkColor = 0
   .AntiAliasing = True
   With .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      With .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = RGB(255,0,0)
         .LinkToWidth("LA2") = 2
         .LinkToPen("LA2") = exPenDashDot
         .LinkToShowDir("LA2") = True
      End With
      .Add "L2_A","LA","LA2"
```

```
        .Add "L2_B1","LB","LB21"
        .Add "L2_B2","LB","LB22"
        .Add "L2_B3","LB","LB23"
    End With
End With
```

The following VB.NET sample shows the direction for an arbitrary link only:

```
With AxChartView1
    .ShowLinksDir = False
    .PenWidthLink = 2
    .LinkColor = Color.FromArgb(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",0,"LA").LinkTo = "LB"
        With .Add("L1_B<br><br>Cust",0,"LB")
            .LinkTo = "LA2"
            .LinkToColor("LA2") = 255
            .LinkToWidth("LA2") = 2
            .LinkToPen("LA2") = EXORGCHARTLib.PenTypeEnum.exPenDashDot
            .LinkToShowDir("LA2") = True
        End With
        .Add "L2_A","LA","LA2"
        .Add "L2_B1","LB","LB21"
        .Add "L2_B2","LB","LB22"
        .Add "L2_B3","LB","LB23"
    End With
End With
```

The following C++ sample shows the direction for an arbitrary link only:

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

    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
```

```
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->PutShowLinksDir(VARIANT_FALSE);
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(0);
spChartView1->PutAntiAliasing(VARIANT_TRUE);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"L1_A",0,"LA",vtMissing,vtMissing)->PutLinkTo("LB");
   EXORGCHARTLib::INodePtr var_Node = var_Nodes->Add(L"L1_B<br>
<br>Cust",0,"LB",vtMissing,vtMissing);
      var_Node->PutLinkTo("LA2");
      var_Node->PutLinkToColor(L"LA2",RGB(255,0,0));
      var_Node->PutLinkToWidth(L"LA2",2);
      var_Node->PutLinkToPen(L"LA2",EXORGCHARTLib::exPenDashDot);
      var_Node->PutLinkToShowDir(L"LA2",VARIANT_TRUE);
   var_Nodes->Add(L"L2_A","LA","LA2",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_B1","LB","LB21",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_B2","LB","LB22",vtMissing,vtMissing);
   var_Nodes->Add(L"L2_B3","LB","LB23",vtMissing,vtMissing);
```

The following C# sample shows the direction for an arbitrary link only:

```
axChartView1.ShowLinksDir = false;
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
axChartView1.AntiAliasing = true;
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   var_Nodes.Add("L1_A",0,"LA",null,null).LinkTo = "LB";
   EXORGCHARTLib.Node var_Node = var_Nodes.Add("L1_B<br>
<br>Cust",0,"LB",null,null);
      var_Node.LinkTo = "LA2";
      var_Node.set_LinkToColor("LA2",255);
      var_Node.set_LinkToWidth("LA2",2);
      var_Node.set_LinkToPen("LA2",EXORGCHARTLib.PenTypeEnum.exPenDashDot);
      var_Node.set_LinkToShowDir("LA2",true);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
```

```
var_Nodes.Add("L2_B1","LB","LB21",null,null);
var_Nodes.Add("L2_B2","LB","LB22",null,null);
var_Nodes.Add("L2_B3","LB","LB23",null,null);
```

The following VFP sample shows the direction for an arbitrary link only:

```
with thisform.ChartView1
   .ShowLinksDir = .F.
   .PenWidthLink = 2
   .LinkColor = 0
   .AntiAliasing = .T.
   with .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      with .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = RGB(255,0,0)
         .LinkToWidth("LA2") = 2
         .LinkToPen("LA2") = 3
         .LinkToShowDir("LA2") = .T.
      endwith
      .Add("L2_A","LA","LA2")
      .Add("L2_B1","LB","LB21")
      .Add("L2_B2","LB","LB22")
      .Add("L2_B3","LB","LB23")
   endwith
endwith
```

The following Delphi sample shows the direction for an arbitrary link only:

```
with AxChartView1 do
begin
   ShowLinksDir := False;
   PenWidthLink := 2;
   LinkColor := Color.FromArgb(0,0,0);
   AntiAliasing := True;
   with Nodes do
   begin
      Add('L1_A',TObject(0),'LA',Nil,Nil).LinkTo := 'LB';
```

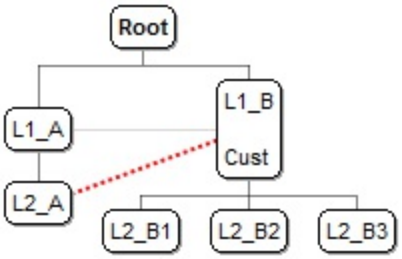
```
    with Add('L1_B<br><br>Cust',TObject(0),'LB',Nil,Nil) do
    begin
      LinkTo := 'LA2';
      LinkToColor['LA2'] := 255;
      LinkToWidth['LA2'] := 2;
      LinkToPen['LA2'] := EXORGCHARTLib.PenTypeEnum.exPenDashDot;
      LinkToShowDir['LA2'] := True;
    end;
    Add('L2_A','LA','LA2',Nil,Nil);
    Add('L2_B1','LB','LB21',Nil,Nil);
    Add('L2_B2','LB','LB22',Nil,Nil);
    Add('L2_B3','LB','LB23',Nil,Nil);
  end;
end
```

property Node.LinkToWidth(Key as String) as Long

Specifies the width to display the LinkTo line.

Type	Description
Key as String	A String expression that specifies the key of the node to link to.
Long	A long expression that specifies the width to show the link.

The LinkToWidth property specifies the width for an linkto line. Use the LinkTo property to arbitrary link the current nodes with others. The PenWidthLinkTo property specifies the width of the links between nodes. The LinkToColor, LinkToPen, LinkToRound, LinkToShowDir property specifies the color, pen, roundness, direction to show an arbitrary link.



The following VB sample changes the width for an arbitrary link:

```
With ChartView1
   .PenWidthLink = 2
   .LinkColor = 0
   With .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      With .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = RGB(255,0,0)
         .LinkToWidth("LA2") = 2
      End With
      .Add "L2_A","LA","LA2"
      .Add "L2_B1","LB","LB21"
      .Add "L2_B2","LB","LB22"
      .Add "L2_B3","LB","LB23"
   End With
End With
```

The following VB.NET sample changes the width for an arbitrary link:

```
With AxChartView1
   .PenWidthLink = 2
   .LinkColor = Color.FromArgb(0,0,0)
   With .Nodes
      .Add("L1_A",0,"LA").LinkTo = "LB"
      With .Add("L1_B<br><br>Cust",0,"LB")
         .LinkTo = "LA2"
         .LinkToColor("LA2") = 255
         .LinkToWidth("LA2") = 2
      End With
      .Add "L2_A","LA","LA2"
      .Add "L2_B1","LB","LB21"
      .Add "L2_B2","LB","LB22"
      .Add "L2_B3","LB","LB23"
   End With
End With
```

The following C++ sample changes the width for an arbitrary link:

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

   #import <ExOrgChart.dll>
   using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(0);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   var_Nodes->Add(L"L1_A",0,"LA",vtMissing,vtMissing)->PutLinkTo("LB");
   EXORGCHARTLib::INodePtr var_Node = var_Nodes->Add(L"L1_B<br>
<br>Cust",0,"LB",vtMissing,vtMissing);
      var_Node->PutLinkTo("LA2");
```

```
        var_Node->PutLinkToColor(L"LA2",RGB(255,0,0));
        var_Node->PutLinkToWidth(L"LA2",2);
    var_Nodes->Add(L"L2_A","LA","LA2",vtMissing,vtMissing);
    var_Nodes->Add(L"L2_B1","LB","LB21",vtMissing,vtMissing);
    var_Nodes->Add(L"L2_B2","LB","LB22",vtMissing,vtMissing);
    var_Nodes->Add(L"L2_B3","LB","LB23",vtMissing,vtMissing);
```

The following C# sample changes the width for an arbitrary link:

```
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
    var_Nodes.Add("L1_A",0,"LA",null,null).LinkTo = "LB";
    EXORGCHARTLib.Node var_Node = var_Nodes.Add("L1_B<br>
<br>Cust",0,"LB",null,null);
        var_Node.LinkTo = "LA2";
        var_Node.set_LinkToColor("LA2",255);
        var_Node.set_LinkToWidth("LA2",2);
    var_Nodes.Add("L2_A","LA","LA2",null,null);
    var_Nodes.Add("L2_B1","LB","LB21",null,null);
    var_Nodes.Add("L2_B2","LB","LB22",null,null);
    var_Nodes.Add("L2_B3","LB","LB23",null,null);
```

The following VFP sample changes the width for an arbitrary link:

```
with thisform.ChartView1
    .PenWidthLink = 2
    .LinkColor = 0
    with .Nodes
        .Add("L1_A",0,"LA").LinkTo = "LB"
        with .Add("L1_B<br><br>Cust",0,"LB")
            .LinkTo = "LA2"
            .LinkToColor("LA2") = RGB(255,0,0)
            .LinkToWidth("LA2") = 2
        endwith
        .Add("L2_A","LA","LA2")
        .Add("L2_B1","LB","LB21")
        .Add("L2_B2","LB","LB22")
```

```
    .Add("L2_B3","LB","LB23")
  endwith
endwith
```

The following Delphi sample changes the width for an arbitrary link:

```
with AxChartView1 do
begin
  PenWidthLink := 2;
  LinkColor := Color.FromArgb(0,0,0);
  with Nodes do
  begin
    Add('L1_A',TObject(0),'LA',Nil,Nil).LinkTo := 'LB';
    with Add('L1_B<br><br>Cust',TObject(0),'LB',Nil,Nil) do
    begin
      LinkTo := 'LA2';
      LinkToColor['LA2'] := 255;
      LinkToWidth['LA2'] := 2;
    end;
    Add('L2_A','LA','LA2',Nil,Nil);
    Add('L2_B1','LB','LB21',Nil,Nil);
    Add('L2_B2','LB','LB22',Nil,Nil);
    Add('L2_B3','LB','LB23',Nil,Nil);
  end;
end
```

property Node.NextNode as Node

Gets the next sibling node.

Type	Description
Node	A Node object that identifies the next sibling node.

Use the NextNode property to determine the next sibling node. The NextNode property gets nothing, if not next node is found. Use the PrevNode property to determine the previous sibling node. Use the LastNode property to determine the last child node. Use the FirstNode property to get the first child node. Use the NodeCount property to get the number of child nodes. Use the Position property to change the node's position. Use the Nodes property to access the nodes collection. Use the Add method to add a child node.

The following VB sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLibCtl.Node)
   Dim c As EXORGCHARTLibCtl.Node
   Set c = n.FirstNode
   While Not c Is Nothing
      Debug.Print c.Caption
      enumRec c
      Set c = c.NextNode
   Wend
End Sub
```

The following C++ sample enumerates recursively all the child nodes:

```
void enumRec( CNode* pNode )
{
   if ( pNode != NULL )
   {
      CNode child = pNode->GetFirstNode();
      while ( child.m_lpDispatch != NULL )
      {
         OutputDebugString( child.GetCaption() );
         OutputDebugString( "\r\n" );
         enumRec( &child );
         child = child.GetNextNode();
      }
```

```
    }
}
```

The following VB.NET sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLib.Node)
    Dim c As EXORGCHARTLib.Node = n.FirstNode
    While Not c Is Nothing
        Debug.Print(c.Caption)
        enumRec(c)
        c = c.NextNode
    End While
End Sub
```

The following C# sample enumerates recursively all the child nodes:

```
private void enumRec(EXORGCHARTLib.Node node)
{
    EXORGCHARTLib.Node child = node.FirstNode;
    while (child != null)
    {
        System.Diagnostics.Debug.WriteLine(child.Caption);
        enumRec(child);
        child = child.NextNode;
    }
}
```

The following VFP sample enumerates recursively all the child nodes:

```
LPARAMETERS node

local child
child = node.FirstNode
do while ( !isnull( child ) )
    wait window nowait child.Caption
    thisform.enumrec( child )
    child = child.NextNode
enddo
```

property Node.NodeCount as Long

Retrieves the number of child nodes.

Type	Description
Long	A long expression that indicates the number of child nodes.

Use the NodeCount property to determine the number of child nodes. Use the FirstNode property to determine the first child node. Use the NextNode property to determine the next sibling node. Use the Position property to change the node's position. Use the LastNode property to determine the last child node. Use the Nodes property to access the nodes collection. Use the Add method to add a child node. Use the Item property to retrieve a node by its key.

The following VB sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLibCtl.Node)
   Dim c As EXORGCHARTLibCtl.Node
   Set c = n.FirstNode
   While Not c Is Nothing
      Debug.Print c.Caption
      enumRec c
      Set c = c.NextNode
   Wend
End Sub
```

The following C++ sample enumerates recursively all the child nodes:

```
void enumRec( CNode* pNode )
{
   if ( pNode != NULL )
   {
      CNode child = pNode->GetFirstNode();
      while ( child.m_lpDispatch != NULL )
      {
         OutputDebugString( child.GetCaption() );
         OutputDebugString( "\r\n" );
         enumRec( &child );
         child = child.GetNextNode();
      }
```

```
    }
}
```

The following VB.NET sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLib.Node)
    Dim c As EXORGCHARTLib.Node = n.FirstNode
    While Not c Is Nothing
        Debug.Print(c.Caption)
        enumRec(c)
        c = c.NextNode
    End While
End Sub
```

The following C# sample enumerates recursively all the child nodes:

```
private void enumRec(EXORGCHARTLib.Node node)
{
    EXORGCHARTLib.Node child = node.FirstNode;
    while (child != null)
    {
        System.Diagnostics.Debug.WriteLine(child.Caption);
        enumRec(child);
        child = child.NextNode;
    }
}
```

The following VFP sample enumerates recursively all the child nodes:

```
LPARAMETERS node

local child
child = node.FirstNode
do while ( !isnull( child ) )
    wait window nowait child.Caption
    thisform.enumrec( child )
    child = child.NextNode
enddo
```

property Node.Nodes as IUnknown FAR*

Gets the collection of nodes.

Type	Description
IUnknown FAR*	An Object that implements the IEnumVARIANT interface.

Only for internal use.

property Node.Padding(Edge as PaddingEdgeEnum) as Long

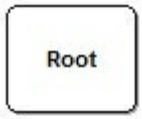
Returns or sets a value that indicates the padding of the node.

Type	Description
Edge as PaddingEdgeEnum	A PaddingEdgeEnum expression that specifies the edge to be updated / requested
Long	A long expression that defines the node's padding

Use the Padding property of the Node to define the padding for specified node. The DefaultNodePadding property defines the padding for all nodes. The BackColor property defines the node's background color / EBN object. Use the FixedHeight and FixedWidth properties to define the size of a specified node. The BackgroundExt property provides unlimited options to add more colors, patterns, text, icons, pictures, frames to any node.

The following screen shot defines a node with no padding (by default):

The following screen shot defines a node with padding:



The following samples show how you can define the node's padding:

VBA (MS Access, Excell...)

```
With ChartView1
   .BeginUpdate
   .IndentSiblingY = 30
   .ShowLinksDir = True
   .PenWidthLink = 2
   .LinkColor = RGB(0,0,0)
   .AntiAliasing = True
   With .Nodes
      With .Add("L1 A1",,"LA")
         .BackColor = RGB(255,0,0)
         .Padding(-1) = 16
      End With
```

```
        .Add "L1 B1",,"LB"
        .Add "L2 A1","LA","LA2"
        .Add "L2 B2","LB","LB2"
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate
End With
```

VB6

```
With ChartView1
    .BeginUpdate
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
        With .Add("L1 A1",,"LA")
            .BackColor = RGB(255,0,0)
            .Padding(exPaddingAll) = 16
        End With
        .Add "L1 B1",,"LB"
        .Add "L2 A1","LA","LA2"
        .Add "L2 B2","LB","LB2"
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate
End With
```

VB.NET

```
With Exchartview1
    .BeginUpdate()
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = Color.FromArgb(0,0,0)
```

```
    .AntiAliasing = True
    With .Nodes
        With .Add("L1 A1",,"LA")
            .BackColor = Color.FromArgb(255,0,0)
            .set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,16)
        End With
        .Add("L1 B1",,"LB")
        .Add("L2 A1","LA","LA2")
        .Add("L2 B2","LB","LB2")
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxChartView1
    .BeginUpdate()
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
        With .Add("L1 A1",,"LA")
            .BackColor = RGB(255,0,0)
            .Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll) = 16
        End With
        .Add("L1 B1",,"LB")
        .Add("L2 A1","LA","LA2")
        .Add("L2 B2","LB","LB2")
    End With
    .Nodes.Item("root").Caption = "Ls As"
    .EndUpdate()
End With
```

C++

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

    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->BeginUpdate();
spChartView1->PutIndentSiblingY(30);
spChartView1->PutShowLinksDir(VARIANT_TRUE);
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(RGB(0,0,0));
spChartView1->PutAntiAliasing(VARIANT_TRUE);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
   EXORGCHARTLib::INodePtr var_Node = var_Nodes->Add(L"L1
A1",vtMissing,"LA",vtMissing,vtMissing);
      var_Node->PutBackColor(RGB(255,0,0));
      var_Node->PutPadding(EXORGCHARTLib::exPaddingAll,16);
   var_Nodes->Add(L"L1 B1",vtMissing,"LB",vtMissing,vtMissing);
   var_Nodes->Add(L"L2 A1","LA","LA2",vtMissing,vtMissing);
   var_Nodes->Add(L"L2 B2","LB","LB2",vtMissing,vtMissing);
spChartView1->GetNodes()->GetItem("root")->PutCaption(L"Ls As");
spChartView1->EndUpdate();
```

C++ Builder

```
ChartView1->BeginUpdate();
ChartView1->IndentSiblingY = 30;
ChartView1->ShowLinksDir = true;
ChartView1->PenWidthLink = 2;
ChartView1->LinkColor = RGB(0,0,0);
ChartView1->AntiAliasing = true;
Exorgchartlib_tlb::INodesPtr var_Nodes = ChartView1->Nodes;
```

```
   Exorgchartlib_tlb::INodePtr var_Node = var_Nodes->Add(L"L1
A1",TNoParam(),TVariant("LA"),TNoParam(),TNoParam());
      var_Node->BackColor = RGB(255,0,0);
      var_Node-
>set_Padding(Exorgchartlib_tlb::PaddingEdgeEnum::exPaddingAll,16);
   var_Nodes->Add(L"L1 B1",TNoParam(),TVariant("LB"),TNoParam(),TNoParam());
   var_Nodes->Add(L"L2 A1",TVariant("LA"),TVariant("LA2"),TNoParam(),TNoParam());
   var_Nodes->Add(L"L2 B2",TVariant("LB"),TVariant("LB2"),TNoParam(),TNoParam());
ChartView1->Nodes->get_Item(TVariant("root"))->Caption = L"Ls As";
ChartView1->EndUpdate();
```

C#

```
exchartview1.BeginUpdate();
exchartview1.IndentSiblingY = 30;
exchartview1.ShowLinksDir = true;
exchartview1.PenWidthLink = 2;
exchartview1.LinkColor = Color.FromArgb(0,0,0);
exchartview1.AntiAliasing = true;
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
   exontrol.EXORGCHARTLib.Node var_Node = var_Nodes.Add("L1
A1",null,"LA",null,null);
      var_Node.BackColor = Color.FromArgb(255,0,0);

var_Node.set_Padding(exontrol.EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,16)

   var_Nodes.Add("L1 B1",null,"LB",null,null);
   var_Nodes.Add("L2 A1","LA","LA2",null,null);
   var_Nodes.Add("L2 B2","LB","LB2",null,null);
exchartview1.Nodes["root"].Caption = "Ls As";
exchartview1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
```

```
id="ChartView1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   ChartView1.BeginUpdate();
   ChartView1.IndentSiblingY = 30;
   ChartView1.ShowLinksDir = true;
   ChartView1.PenWidthLink = 2;
   ChartView1.LinkColor = 0;
   ChartView1.AntiAliasing = true;
   var var_Nodes = ChartView1.Nodes;
      var var_Node = var_Nodes.Add("L1 A1",null,"LA",null,null);
         var_Node.BackColor = 255;
         var_Node.Padding(-1) = 16;
      var_Nodes.Add("L1 B1",null,"LB",null,null);
      var_Nodes.Add("L2 A1","LA","LA2",null,null);
      var_Nodes.Add("L2 B2","LB","LB2",null,null);
   ChartView1.Nodes.Item("root").Caption = "Ls As";
   ChartView1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:F4DFE455-01FE-420E-A088-64346DCC3791"
id="ChartView1">></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With ChartView1
      .BeginUpdate
      .IndentSiblingY = 30
      .ShowLinksDir = True
```

```
      .PenWidthLink = 2
      .LinkColor = RGB(0,0,0)
      .AntiAliasing = True
      With .Nodes
         With .Add("L1 A1",,"LA")
            .BackColor = RGB(255,0,0)
            .Padding(-1) = 16
         End With
         .Add "L1 B1",,"LB"
         .Add "L2 A1","LA","LA2"
         .Add "L2 B2","LB","LB2"
      End With
      .Nodes.Item("root").Caption = "Ls As"
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axChartView1.BeginUpdate();
axChartView1.IndentSiblingY = 30;
axChartView1.ShowLinksDir = true;
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
axChartView1.AntiAliasing = true;
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   EXORGCHARTLib.Node var_Node = var_Nodes.Add("L1 A1",null,"LA",null,null);
      var_Node.BackColor = (uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0));
      var_Node.set_Padding(EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll,16);
   var_Nodes.Add("L1 B1",null,"LB",null,null);
   var_Nodes.Add("L2 A1","LA","LA2",null,null);
   var_Nodes.Add("L2 B2","LB","LB2",null,null);
axChartView1.Nodes["root"].Caption = "Ls As";
axChartView1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Node,com_Node1,com_Nodes;
   anytype var_Node,var_Node1,var_Nodes;
   ;

   super();

   exchartview1.BeginUpdate();
   exchartview1.IndentSiblingY(30);
   exchartview1.ShowLinksDir(true);
   exchartview1.PenWidthLink(2);
   exchartview1.LinkColor(WinApi::RGB2int(0,0,0));
   exchartview1.AntiAliasing(true);
   var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
      var_Node = com_Nodes.Add("L1 A1",,"LA"); com_Node = var_Node;
         com_Node.BackColor(WinApi::RGB2int(255,0,0));
         com_Node.Padding(-1/*exPaddingAll*/,16);
      com_Nodes.Add("L1 B1",,"LB");
      com_Nodes.Add("L2 A1","LA","LA2");
      com_Nodes.Add("L2 B2","LB","LB2");
   var_Node1 = COM::createFromObject(exchartview1.Nodes()).Item("root");
com_Node1 = var_Node1;
   com_Node1.Caption("Ls As");
   exchartview1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
   BeginUpdate();
   IndentSiblingY := 30;
   ShowLinksDir := True;
```

```
  PenWidthLink := 2;
  LinkColor := Color.FromArgb(0,0,0);
  AntiAliasing := True;
  with Nodes do
  begin
    with Add('L1 A1',Nil,'LA',Nil,Nil) do
    begin
      BackColor := $ff;
      Padding[EXORGCHARTLib.PaddingEdgeEnum.exPaddingAll] := 16;
    end;
    Add('L1 B1',Nil,'LB',Nil,Nil);
    Add('L2 A1','LA','LA2',Nil,Nil);
    Add('L2 B2','LB','LB2',Nil,Nil);
  end;
  Nodes.Item['root'].Caption := 'Ls As';
  EndUpdate();
end
```

Delphi (standard)

```
with ChartView1 do
begin
  BeginUpdate();
  IndentSiblingY := 30;
  ShowLinksDir := True;
  PenWidthLink := 2;
  LinkColor := RGB(0,0,0);
  AntiAliasing := True;
  with Nodes do
  begin
    with Add('L1 A1',Null,'LA',Null,Null) do
    begin
      BackColor := $ff;
      Padding[EXORGCHARTLib_TLB.exPaddingAll] := 16;
    end;
    Add('L1 B1',Null,'LB',Null,Null);
    Add('L2 A1','LA','LA2',Null,Null);
```

```
    Add('L2 B2','LB','LB2',Null,Null);
  end;
  Nodes.Item['root'].Caption := 'Ls As';
  EndUpdate();
end
```

VFP

```
with thisform.ChartView1
  .BeginUpdate
  .IndentSiblingY = 30
  .ShowLinksDir = .T.
  .PenWidthLink = 2
  .LinkColor = RGB(0,0,0)
  .AntiAliasing = .T.
  with .Nodes
    with .Add("L1 A1",Null,"LA")
      .BackColor = RGB(255,0,0)
      .Padding(-1) = 16
    endwith
    .Add("L1 B1",Null,"LB")
    .Add("L2 A1","LA","LA2")
    .Add("L2 B2","LB","LB2")
  endwith
  .Nodes.Item("root").Caption = "Ls As"
  .EndUpdate
endwith
```

dBASE Plus

```
local oChartView,var_Node,var_Nodes

oChartView = form.EXORGCHARTACTIVEXCONTROL1.nativeObject
oChartView.BeginUpdate()
oChartView.IndentSiblingY = 30
oChartView.ShowLinksDir = true
oChartView.PenWidthLink = 2
oChartView.LinkColor = 0x0
```

```
oChartView.AntiAliasing = true
var_Nodes = oChartView.Nodes
   var_Node = var_Nodes.Add("L1 A1",null,"LA")
      var_Node.BackColor = 0xff
      // var_Node.Padding(-1) = 16
      with (oChartView)
         TemplateDef = [dim var_Node]
         TemplateDef = var_Node
         Template = [var_Node.Padding(-1) = 16]
      endwith
   var_Nodes.Add("L1 B1",null,"LB")
   var_Nodes.Add("L2 A1","LA","LA2")
   var_Nodes.Add("L2 B2","LB","LB2")
oChartView.Nodes.Item("root").Caption = "Ls As"
oChartView.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oChartView as P
Dim var_Node as P
Dim var_Nodes as P

oChartView = topparent:CONTROL_ACTIVEX1.activex
oChartView.BeginUpdate()
oChartView.IndentSiblingY = 30
oChartView.ShowLinksDir = .t.
oChartView.PenWidthLink = 2
oChartView.LinkColor = 0
oChartView.AntiAliasing = .t.
var_Nodes = oChartView.Nodes
   var_Node = var_Nodes.Add("L1 A1",,"LA")
      var_Node.BackColor = 255
      ' var_Node.Padding(-1) = 16
      oChartView.TemplateDef = "dim var_Node"
      oChartView.TemplateDef = var_Node
      oChartView.Template = "var_Node.Padding(-1) = 16"
```

```
    var_Nodes.Add("L1 B1",,"LB")
    var_Nodes.Add("L2 A1","LA","LA2")
    var_Nodes.Add("L2 B2","LB","LB2")
oChartView.Nodes.Item("root").Caption = "Ls As"
oChartView.EndUpdate()
```

Visual Objects

```
local var_Node as INode
local var_Nodes as INodes

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:IndentSiblingY := 30
oDCOCX_Exontrol1:ShowLinksDir := true
oDCOCX_Exontrol1:PenWidthLink := 2
oDCOCX_Exontrol1:LinkColor := RGB(0,0,0)
oDCOCX_Exontrol1:AntiAliasing := true
var_Nodes := oDCOCX_Exontrol1:Nodes
    var_Node := var_Nodes:Add("L1 A1",nil,"LA",nil,nil)
        var_Node:BackColor := RGB(255,0,0)
        var_Node:[Padding,exPaddingAll] := 16
    var_Nodes:Add("L1 B1",nil,"LB",nil,nil)
    var_Nodes:Add("L2 A1","LA","LA2",nil,nil)
    var_Nodes:Add("L2 B2","LB","LB2",nil,nil)
oDCOCX_Exontrol1:Nodes:[Item,"root"]:Caption := "Ls As"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oChartView,var_Node,var_Nodes

oChartView = ole_1.Object
oChartView.BeginUpdate()
oChartView.IndentSiblingY = 30
oChartView.ShowLinksDir = true
```

```
oChartView.PenWidthLink = 2
oChartView.LinkColor = RGB(0,0,0)
oChartView.AntiAliasing = true
var_Nodes = oChartView.Nodes
   var_Node = var_Nodes.Add("L1 A1",,"LA")
      var_Node.BackColor = RGB(255,0,0)
      var_Node.Padding(-1,16)
   var_Nodes.Add("L1 B1",,"LB")
   var_Nodes.Add("L2 A1","LA","LA2")
   var_Nodes.Add("L2 B2","LB","LB2")
oChartView.Nodes.Item("root").Caption = "Ls As"
oChartView.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComIndentSiblingY to 30
   Set ComShowLinksDir to True
   Set ComPenWidthLink to 2
   Set ComLinkColor to (RGB(0,0,0))
   Set ComAntiAliasing to True
   Variant voNodes
   Get ComNodes to voNodes
   Handle hoNodes
   Get Create (RefClass(cComNodes)) to hoNodes
   Set pvComObject of hoNodes to voNodes
      Variant voNode
      Get ComAdd of hoNodes "L1 A1" "LA" Nothing Nothing to voNode
      Handle hoNode
      Get Create (RefClass(cComNode)) to hoNode
      Set pvComObject of hoNode to voNode
         Set ComBackColor of hoNode to (RGB(255,0,0))
         Set ComPadding of hoNode OLEexPaddingAll to 16
      Send Destroy to hoNode
```

```
        Get ComAdd of hoNodes "L1 B1" "LB" Nothing Nothing to Nothing
        Get ComAdd of hoNodes "L2 A1" "LA" "LA2" Nothing Nothing to Nothing
        Get ComAdd of hoNodes "L2 B2" "LB" "LB2" Nothing Nothing to Nothing
    Send Destroy to hoNodes
    Variant voNodes1
    Get ComNodes to voNodes1
    Handle hoNodes1
    Get Create (RefClass(cComNodes)) to hoNodes1
    Set pvComObject of hoNodes1 to voNodes1
        Variant voNode1
        Get ComItem of hoNodes1 "root" to voNode1
        Handle hoNode1
        Get Create (RefClass(cComNode)) to hoNode1
        Set pvComObject of hoNode1 to voNode1
            Set ComCaption of hoNode1 to "Ls As"
        Send Destroy to hoNode1
    Send Destroy to hoNodes1
    Send ComEndUpdate
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oChartView
   LOCAL oNode
   LOCAL oNodes

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

```
   oChartView := XbpActiveXControl():new( oForm:drawingArea )
   oChartView:CLSID  := "Exontrol.ChartView.1" /*{F4DFE455-01FE-420E-A088-
64346DCC3791}*/
   oChartView:create(,, {10,60},{610,370} )

      oChartView:BeginUpdate()
      oChartView:IndentSiblingY := 30
      oChartView:ShowLinksDir := .T.
      oChartView:PenWidthLink := 2
      oChartView:SetProperty("LinkColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oChartView:AntiAliasing := .T.
      oNodes := oChartView:Nodes()
         oNode := oNodes:Add("L1 A1",,"LA")
            oNode:SetProperty("BackColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 255,0,0 } )  , .F. ))
            oNode:SetProperty("Padding",-1/*exPaddingAll*/,16)
         oNodes:Add("L1 B1",,"LB")
         oNodes:Add("L2 A1","LA","LA2")
         oNodes:Add("L2 B2","LB","LB2")
      oChartView:Nodes:Item("root"):Caption := "Ls As"
      oChartView:EndUpdate()

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

property Node.Parent as Node

Retrieves or sets the parent of the node.

Type	Description
Node	A Node object that identifies the parent of the node.

Use the Parent property to get the parent of the node. Use the Parent property to move a node from a parent to another. The Parent property gets nothing if the node has no parent. The root node has no parent. Use the Root property to access the root node of the organigram. Use the Item property to access an item/node by its key. Use the FirstNode property to retrieve the first child node. Use the NextNode property to determine the next sibling node. Use the Position property to change the node's position. Use the LastNode property to determine the last child node. Use the Nodes property to access the nodes collection. Use the Add method to add a child node.

The following VB sample changes the node's parent by dragging:

```
Dim n As Node

Private Sub ChartView1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   Set n = ChartView1.NodeFromPoint(-1, -1)
   If Not (n Is Nothing) Then
      If n.IsAssistant Or n.IsGroup Then
         Set n = Nothing
      End If
   End If
End Sub

Private Sub ChartView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   If (Button = 1) Then
      If Not (n Is Nothing) Then
         Dim nNew As Node
         Set nNew = ChartView1.NodeFromPoint(-1, -1)
         If Not (nNew Is Nothing) Then
            If Not nNew.IsAssistant And Not nNew.IsGroup Then
               n.Parent = nNew
```

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

The following VB sample moves the "wnd" node to "gdi" node:

```
With ChartView1.Nodes
  .Item("wnd").Parent = .Item("gdi")
End With
```

When moving a node from a parent to another you need to be sure:

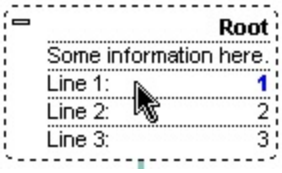
- the new parent node is not child of the current node. For instance, if A is parent of B, and B is parent of C, you can't move the B as being child of C.
- the current node and the new parent node are not assistant nodes. The AddAssistant node adds assistant nodes. An assistant node can't have child nodes. The Add method adds normal nodes, that can move from a parent to another.

property Node.PenBorderNode as PenTypeEnum

Specifies the type of pen used to paint the node's borders.

Type	Description
PenTypeEnum	A PenTypeEnum expression that indicates the type of the pen used to pain the borders of the node.

Use the PenBorderNode property to define the type of the pen used to paint the borders for a specified node. Use the PenBorderNode property to define the type of the pen used to paint the borders for all nodes. Use the ShadowNode property to hide the node's shadow. Use the BackColor property to specify the node's background color. Use the DrawRoundNode property to specify whether the node has a round border. Use the BorderColor property to assign a different color for the node's border.



property Node.Picture as Variant

Retrieves or sets a graphic to be displayed in the node.

Type	Description
Variant	A VARIANT value that indicates the picture in the following formats: • A picture object being displayed in the node (IPicture interface) • A safe array of bytes that indicates the OLE Object field. Use for OLE objects (such as Microsoft Word documents, Microsoft Excel spreadsheets, **pictures**, sounds, or other binary data) that were created in other programs using the OLE protocol. • A string expression that specifies the key of the picture being inserted using the HTMLPicture property • A string expression that indicates the picture's file path • A string expression that indicates the base64 encoded string that holds the picture. Use the eximages tool to save your icons as base64 encoded format.

Use the Picture property to load an user defined picture to a node. Use the Image property to display a 16x16 icon in the node. Use the PictureAlignment property to align the picture in the node. Also, you can assign a picture to a node when calling Add method. Use the BackColor property to specify the node's background color. Use the Picture property to display a picture on the control's background. Use the Caption property to assign a caption to a node. You can insert custom sized pictures inside HTML captions using the <img> built-in HTML format. The Picture property supports: BMP, EMF, EXIF, GIF, ICON, JPEG, PNG, TIFF or WMF formats. The PictureWidth and PictureHeight properties controls the size of displayed picture. The PictureWidthNode and PictureHeightNode properties of the ChartView object handles the size of the displayed pictures for all the nodes.

If using the /NET assembly version, you can pass an System.Drawing.Image object to the Picture property like: .Nodes.Add("Child2", .Root, "21").**Picture** = PictureBox1.**Image**

The size of the picture being shown in the node is computed as follow:

- if the PictureWidth and/or PictureHeight property is specified, it indicates the width or the height of the picture being shown in the node.
- if the PictureWidthNode and/or PictureHeightNode properties of the ChartView object is specified, it indicates the width or the height of the picture being shown in the node.

- If none of these is specified, the node displays the picture using its size.

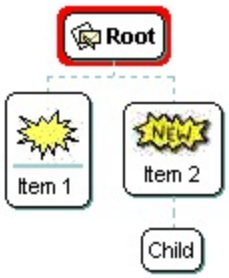
In any case, the size of the picture is also influenced by the the aspect ratio for the node's picture as it is specified by PictureAspectRatio or PictureAspectRatioNode property while any of these is not exAspectRatioNone.

The following VB sample loads icons and pictures using the BASE64 encoded strings:

```
Dim s As String
With ChartView1
   .BackColor = vbWhite

   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl0vmEx

   With .Root
      .Image = 1
   End With

   With .Nodes.Add("Item 1")
      s =
"gBCJr+BAAg0HGwEgwog4jg4ig4BAEFg4AZEKisZjUbAAzg5mg6Zg7Mg7/g0ek8oGcgjsijsk

      s = s +
"XgBadIDXdYSXRb9wWBclK2taF1gAI5HiPaN8oPdINWbaF23KAwyWkNYyXxg9p3WNYjU/c

      .Picture = s
   End With

   With .Nodes.Add("Item 2", , "ketA")
      s =
"gBCJoqBAAg0HGwEgwog4qg4Xg4BAEFg4AegDisZjUbgwzg5mg6Zg7Mg7/jsHGceAAzkEr

      s = s +
"XU5WtD2VbVlFZUwDAAGQD3La4yjIAADAOQQrAOFYc2IGRA1nTwAVgAASXGKwVi7fgADI
```

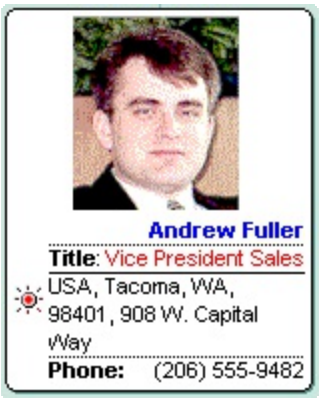
```
        s = s +
"K7lDLYABKAaArVsisUuoWK6m1UKyYIpSVihRrPSIILKQi51OK6USqJUDIlSzBU4LNSKjVDytlll\



        .Picture = s
    End With

    With .Nodes.Add("Child", "ketA")
    End With

End With
```

Run the sample and you get:



The following two VB samples are equivalents:

```
With ChartView1
        With .Nodes
            With .Add("<r><dotline><b><fgcolor=0000FF>Andrew Fuller</fgcolor></b>
<br><solidline><b>Title</b>:<r><fgcolor=FF0000>Vice President Sales</fgcolor>
<br>USA, Tacoma, WA, 98401, 908 W. Capital Way<br><dotline><upline><b>Phone:
</b><r>(206) 555-9482", , , 1, "c:\temp\sample\andrew.gif")
                .BackColor = vbWhite
            End With
        End With
End With
```




```
With ChartView1
    With .Nodes
        With .Add("<r><dotline><b><fgcolor=0000FF>Andrew Fuller</fgcolor></b>
<br><solidline><b>Title</b>:<r><fgcolor=FF0000>Vice President Sales</fgcolor>
<br>USA, Tacoma, WA, 98401, 908 W. Capital Way<br><dotline><upline><b>Phone:
</b><r>(206) 555-9482", , , 1)
            .BackColor = vbWhite
            .Picture = "c:\temp\sample\andrew.gif"
        End With
    End With
  End With
```

I am wondering how to add a bitmap to a node without using an image list? Actually from a CBitmap that I already have. The idea is to get an IPicture object that displays that HBITMAP handle, so we have to use the OleCreatePictureIndirect API function that does the job. We can use the CPictureHolder class, already defined in the afxctl.h file, or we can use a direct function IPictureFromCBitmap as defined bellow.

1. Using the CPictureHolder class

```
#include <afxctl.h>
CPictureHolder pict;
if ( pict.CreateFromBitmap( (HBITMAP)bitmap.Detach() ) )
{
    COleVariant vtPicture;
    V_VT( &vtPicture ) = VT_DISPATCH;
    (V_DISPATCH( &vtPicture ) = pict.GetPictureDispatch() )->AddRef();
    node.SetPicture( vtPicture );
}
```

2. Using the IPictureFromCBitmap function

```
#include <atlbase.h>
CComVariant IPictureFromCBitmap( CBitmap* p )
{
   if ( NULL != p )
   {
      PICTDESC pDesc = {0};
      pDesc.cbSizeofstruct = sizeof( PICTDESC );
      pDesc.picType = PICTYPE_BITMAP;
      pDesc.bmp.hbitmap = (HBITMAP)p->Detach();
      CComPtr<IPicture> spPicture;
      if ( SUCCEEDED( OleCreatePictureIndirect( &pDesc, IID_IPicture, FALSE,
(LPVOID*)&spPicture ) ) )
         return CComVariant( CComQIPtr<IDispatch>( spPicture ) );
   }
   return NULL;
}
```

and you call something like node.SetPicture(IPictureFromCBitmap(&bitmap));

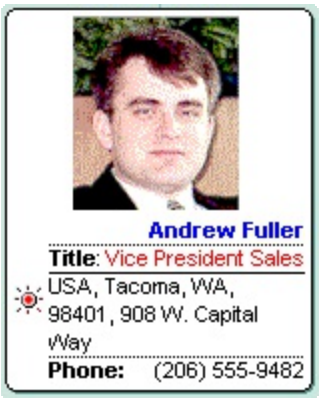
where the node member is the node in the control to put the picture, and the bitmap is an object of CBitmap type.

property Node.PictureAlignment as ImageAlignmentEnum

Specifies the alignment of the picture within the node.

Type	Description
ImageAlignmentEnum	An ImageAlignmentEnum expression that indicates the alignment of the picture.

Use the PictureAlignment property to align the picture in the node. Use the Picture property to load an user defined picture to a node. The PictureAlignment property has no effect if the node has no picture loaded. Use the BackColor property to specify the node's background color. Use the Picture property to display a picture on the control's background.



property Node.PictureAspectRatio as AspectRatioEnum

Specifies the aspect ratio of the node's picture.

Type	Description
AspectRatioEnum	An AspectRatioEnum expression that specifies the aspect ratio for the node's picture.

By default, the PictureAspectRatio property is exAspectRatioNone. Even if the PictureAspectRatio property is not set, the control's PictureAspectRatioNode property may control's the aspect ratio for all pictures. Use the Picture property to specify the node's picture.

The aspect ratio for the node's picture works as follows:

- **exAspectRatioWidth**, the PictureWidth or PictureWidthNode property must specify the fixed width, so the height of the displaying picture is calculated based on the original size, so it keeps its aspect ratio.
- **exAspectRatioHeight**, the PictureHeight or PictureHeightNode property must specify the fixed height, so the width of the displaying picture is calculated based on the original size, so it keeps its aspect ratio.

The size of the picture being shown in the node is computed as follow:

- if the PictureWidth and/or PictureHeight property is specified, it indicates the width or the height of the picture being shown in the node.
- if the PictureWidthNode and/or PictureHeightNode properties of the ChartView object is specified, it indicates the width or the height of the picture being shown in the node.
- If none of these is specified, the node displays the picture using its size.

In any case, the size of the picture is also influenced by the the aspect ratio for the node's picture as it is specified by PictureAspectRatio or PictureAspectRatioNode property while any of these is not exAspectRatioNone.

property Node.PictureHeight as Long

Specifies the height of the node's picture.

Type	Description
Long	A long expression that specifies the height to display the node's picture. If not specified, the picture's height is used instead.

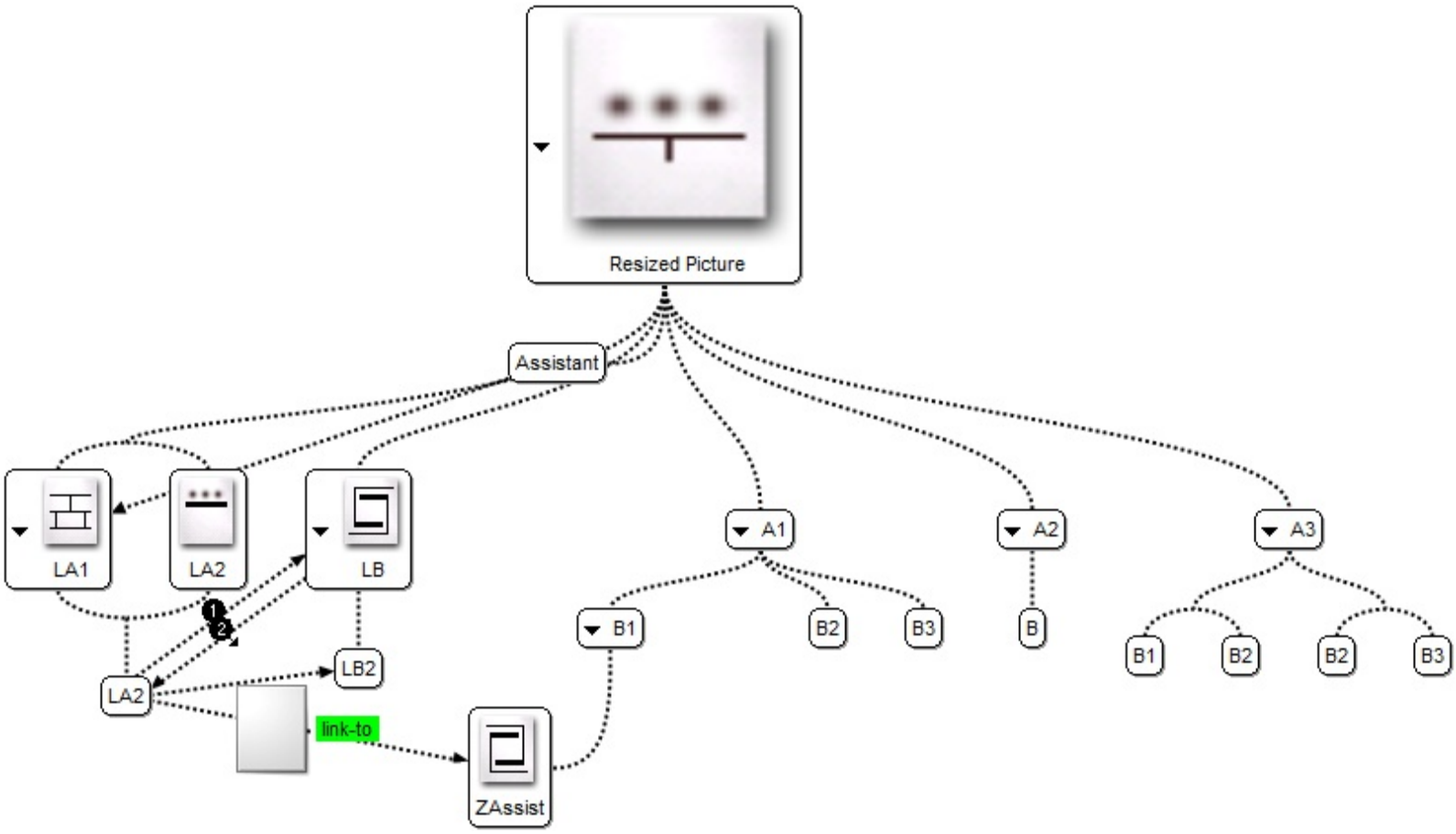
By default, the PictureHeight property is -1. Use the PictureHeight property to specify a different height to display the node's picture. Use the Picture property to specify the node's picture. The PictureWidthNode and PictureHeightNode properties of the ChartView object handles the size of the displayed pictures for all the nodes. The PictureWidth and PictureHeight properties handles the size of displayed picture for specified node. The PictureAspectRatio property specifies the aspect ratio for the node's picture based on the original width or height.

The size of the picture being shown in the node is computed as follow:

- if the PictureWidth and/or PictureHeight property is specified, it indicates the width or the height of the picture being shown in the node.
- if the PictureWidthNode and/or PictureHeightNode properties of the ChartView object is specified, it indicates the width or the height of the picture being shown in the node.
- If none of these is specified, the node displays the picture using its size.

In any case, the size of the picture is also influenced by the the aspect ratio for the node's picture as it is specified by PictureAspectRatio or PictureAspectRatioNode property while any of these is not exAspectRatioNone.

The following screen shot shows the root's picture resized, while the other nodes display the pictures using their original size.


Resized Picture
Assistant
LA1
LA2
LB
A1
A2
A3
B1
B2
B3
B
B1
B2
B2
B3
LA2
LB2
link-to
ZAssist

property Node.PictureWidth as Long

Specifies the width of the node's picture.

Type	Description
Long	A long expression that specifies the width to display the node's picture. If not specified, the picture's width is used instead.

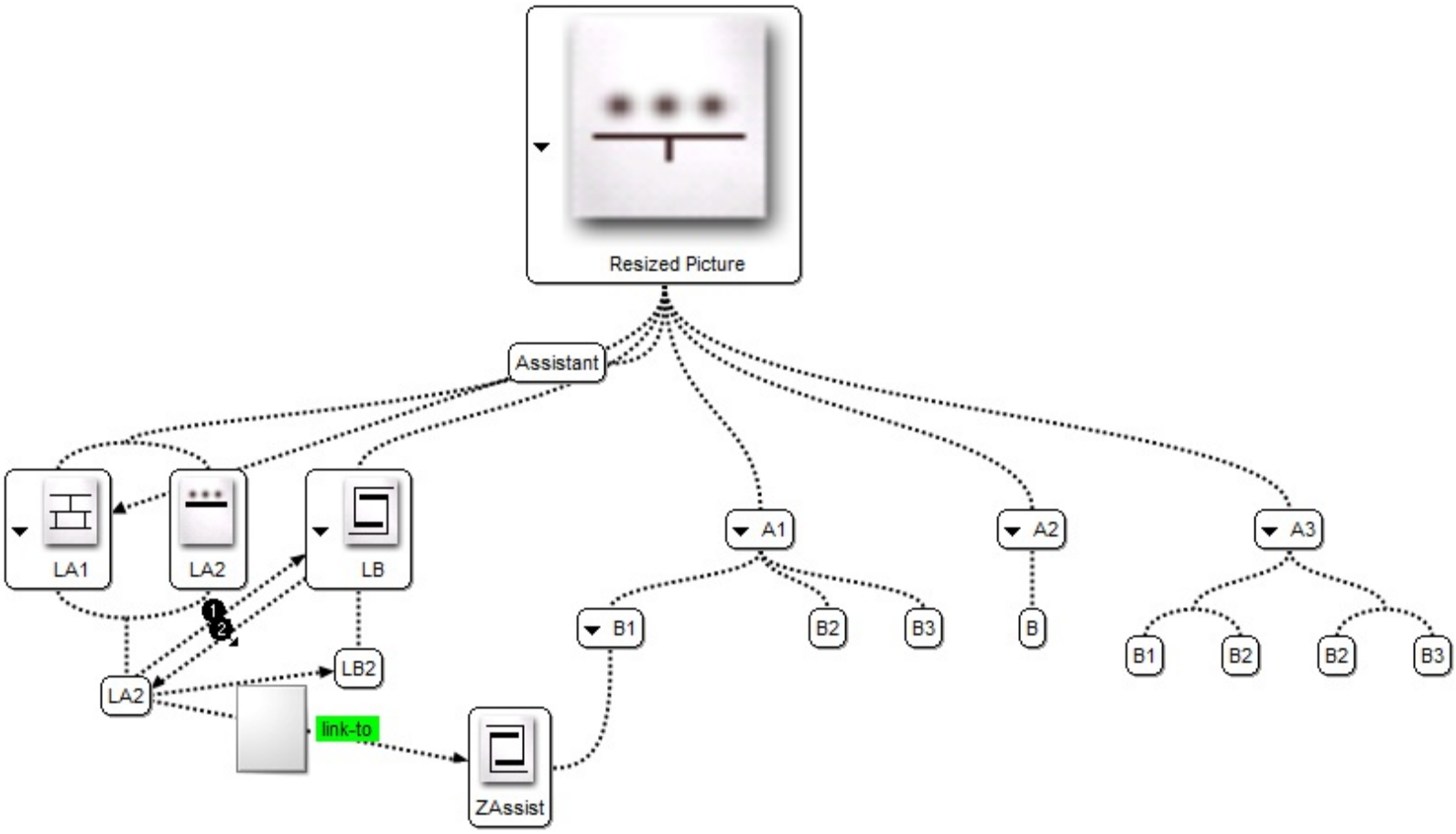
By default, the PictureWidth property is -1. Use the PictureWidth property to specify a different width to display the node's picture. Use the Picture property to specify the node's picture. The PictureWidthNode and PictureHeightNode properties of the ChartView object handles the size of the displayed pictures for all the nodes. The PictureWidth and PictureHeight properties handles the size of displayed picture for specified node. The PictureAspectRatio property specifies the aspect ratio for the node's picture based on the original width or height.

The size of the picture being shown in the node is computed as follow:

- if the PictureWidth and/or PictureHeight property is specified, it indicates the width or the height of the picture being shown in the node.
- if the PictureWidthNode and/or PictureHeightNode properties of the ChartView object is specified, it indicates the width or the height of the picture being shown in the node.
- If none of these is specified, the node displays the picture using its size.

In any case, the size of the picture is also influenced by the the aspect ratio for the node's picture as it is specified by PictureAspectRatio or PictureAspectRatioNode property while any of these is not exAspectRatioNone.

The following screen shot shows the root's picture resized, while the other nodes display the pictures using their original size.


Resized Picture
Assistant
LA1
LA2
LB
A1
A2
A3
B1
B2
B3
B
B1
B2
B2
B3
LA2
LB2
link-to
ZAssist

property Node.Position as Long

Specifies the position of the node.

Type	Description
Long	A long expression that specifies the position of the node.

Use the Position property to determine the position of the node in the child nodes collection of the parent node. Use the Index or Key property to identify a node. Use the FirstNode property to get the first child node. Use the NextNode property to determine the next sibling node. Use the NodeCount property to get the number of child nodes. Use the LastNode property to determine the last child node. Use the Nodes property to access the nodes collection. Use the Add method to add a child node. Use the Root property to get the root node. Use the Caption property to specify the caption of the node.

The following VB sample displays the position for each child node that belongs to the root node:

```
With ChartView1
    With .Root
      Dim c As EXORGCHARTLibCtl.Node
      Set c = .FirstNode
      While Not c Is Nothing
        Debug.Print c.Position
        Set c = c.NextNode
      Wend
    End With
End With
```

The following VB sample changes the position of the last visible child node that belongs to the root node:

```
With ChartView1.Root
  .LastNode.Position = 0
End With
```

The following C++ sample changes the position of the last visible child node that belongs to the root node:

```
m_chartview.GetRoot().GetLastNode().SetPosition(0);
```

The following VB.NET sample changes the position of the last visible child node that belongs to the root node:

```
With AxChartView1.Root
    .LastNode.Position = 0
End With
```

The following C# sample changes the position of the last visible child node that belongs to the root node:

```
axChartView1.Root.LastNode.Position = 0;
```

The following VFP sample changes the position of the last visible child node that belongs to the root node:

```
With thisform.ChartView1
    .Root.LastNode.Position = 0
EndWith
```

property Node.PrevNode as Node

Gets the previous sibling node.

Type	Description
Node	A Node object that identifies the previous sibling node.

Use the PrevNode property to determine the previous sibling node. Use the NextNode property to determine the next sibling node Use the LastNode property to determine the last child node. Use the FirstNode property to get the first child node. Use the NodeCount property to get the number of child nodes. Use the Position property to change the node's position.

The following VB sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLibCtl.Node)
   Dim c As EXORGCHARTLibCtl.Node
   Set c = n.FirstNode
   While Not c Is Nothing
      Debug.Print c.Caption
      enumRec c
      Set c = c.NextNode
   Wend
End Sub
```

The following C++ sample enumerates recursively all the child nodes:

```
void enumRec( CNode* pNode )
{
   if ( pNode != NULL )
   {
      CNode child = pNode->GetFirstNode();
      while ( child.m_lpDispatch != NULL )
      {
         OutputDebugString( child.GetCaption() );
         OutputDebugString( "\r\n" );
         enumRec( &child );
         child = child.GetNextNode();
      }
   }
```

```
}
```

The following VB.NET sample enumerates recursively all the child nodes:

```
Private Sub enumRec(ByVal n As EXORGCHARTLib.Node)
   Dim c As EXORGCHARTLib.Node = n.FirstNode
   While Not c Is Nothing
      Debug.Print(c.Caption)
      enumRec(c)
      c = c.NextNode
   End While
End Sub
```

The following C# sample enumerates recursively all the child nodes:

```
private void enumRec(EXORGCHARTLib.Node node)
{
   EXORGCHARTLib.Node child = node.FirstNode;
   while (child != null)
   {
      System.Diagnostics.Debug.WriteLine(child.Caption);
      enumRec(child);
      child = child.NextNode;
   }
}
```

The following VFP sample enumerates recursively all the child nodes:

```
LPARAMETERS node

local child
child = node.FirstNode
do while ( !isnull( child ) )
   wait window nowait child.Caption
   thisform.enumrec( child )
   child = child.NextNode
enddo
```

method Node.Remove ()

Removes recursively the child nodes, the assistant nodes and all the nodes in the same group, and if possible remove the node itself.

Type	Description

method Node.RemoveAssistant (Index as Variant)

Removes an assistant node.

Type	Description
Index as Variant	A long expression that indicates the index of the assistant node being removed.

Use the RemoveAssistant method to remove an assistant node. The RemoveAssistant method removes only the nodes being added using the AddAssistant method. The IsAssistant property determines whether a node is an assistant node or a child node. Use the ClearAssistants method to clear the assistant nodes collection. Use the ShowAssistants property to hide all assistant nodes. Use the Remove method to remove a child node.

method Node.RemoveGroup (Index as Variant)

Removes a node from the group.

Type	Description
Index as Variant	A long expression that specifies the index of the group node being removed.

Use the RemoveGroup method to remove a group node. Use the AddGroup method to add new nodes in the same group (multiple parents). Use the IsGroup property to specify whether a node was added using the AddGroup method. Use the CountGroup and Group properties to access or enumerate the group nodes collection. Use the InflateGroupY property to specify the indentation of the group on vertical axis.

property Node.ShadowNode as Boolean

Specifies whether the node has shadow.

Type	Description
Boolean	A boolean expression that indicates whether the node has shadow.

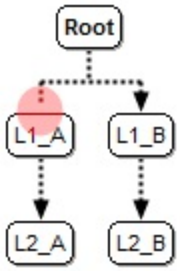
Use the ShadowNode property to hide the shadow for a specific node. The ShadowNode property determines whether the control displays a shadow for nodes. Use the PenBorderNode property to define the type of the pen used to paint the borders for a specified node. Use the SelColor property to hide the mark around the selected node. Use the SelectNode property to specify the selected node.

property Node.ShowLinkDir as Boolean

Shows or hides the direction between the node and its parent.

Type	Description
Boolean	A Boolean expression that indicates whether the direction of the link between the parent and the current node is shown or hidden.

By default, the ShowLinkDir property is True. This property has effect only if the ShowLinksDir property of the ChartView is True. You can use the ShowLinkDir property to hide specific arrows in the chart.



The following samples shows how to hide directions for specific links:

VBA (MS Access, Excell...)

```
With ChartView1
   .IndentSiblingY = 30
   .ShowLinksDir = True
   .PenWidthLink = 2
   .LinkColor = RGB(0,0,0)
   .AntiAliasing = True
   With .Nodes
      .Add("L1_A",0,"LA").ShowLinkDir = False
      .Add "L1_B",0,"LB"
      .Add "L2_A","LA","LA2"
      .Add "L2_B","LB","LB2"
   End With
End With
```

VB6

```
With ChartView1
```

```
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",0,"LA").ShowLinkDir = False
        .Add "L1_B",0,"LB"
        .Add "L2_A","LA","LA2"
        .Add "L2_B","LB","LB2"
    End With
End With
```

VB.NET

```
With Exchartview1
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = Color.FromArgb(0,0,0)
    .AntiAliasing = True
    With .Nodes
        .Add("L1_A",0,"LA").ShowLinkDir = False
        .Add("L1_B",0,"LB")
        .Add("L2_A","LA","LA2")
        .Add("L2_B","LB","LB2")
    End With
End With
```

VB.NET for /COM

```
With AxChartView1
    .IndentSiblingY = 30
    .ShowLinksDir = True
    .PenWidthLink = 2
    .LinkColor = RGB(0,0,0)
    .AntiAliasing = True
    With .Nodes
```

```
        .Add("L1_A",0,"LA").ShowLinkDir = False
        .Add("L1_B",0,"LB")
        .Add("L2_A","LA","LA2")
        .Add("L2_B","LB","LB2")
    End With
End With
```

C++

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

    #import <ExOrgChart.dll>
    using namespace EXORGCHARTLib;
*/
EXORGCHARTLib::IChartViewPtr spChartView1 = GetDlgItem(IDC_CHARTVIEW1)-
>GetControlUnknown();
spChartView1->PutIndentSiblingY(30);
spChartView1->PutShowLinksDir(VARIANT_TRUE);
spChartView1->PutPenWidthLink(2);
spChartView1->PutLinkColor(RGB(0,0,0));
spChartView1->PutAntiAliasing(VARIANT_TRUE);
EXORGCHARTLib::INodesPtr var_Nodes = spChartView1->GetNodes();
    var_Nodes->Add(L"L1_A",long(0),"LA",vtMissing,vtMissing)-
>PutShowLinkDir(VARIANT_FALSE);
    var_Nodes->Add(L"L1_B",long(0),"LB",vtMissing,vtMissing);
    var_Nodes->Add(L"L2_A","LA","LA2",vtMissing,vtMissing);
    var_Nodes->Add(L"L2_B","LB","LB2",vtMissing,vtMissing);
```

C#

```
exchartview1.IndentSiblingY = 30;
exchartview1.ShowLinksDir = true;
exchartview1.PenWidthLink = 2;
exchartview1.LinkColor = Color.FromArgb(0,0,0);
```

```
exchartview1.AntiAliasing = true;
exontrol.EXORGCHARTLib.Nodes var_Nodes = exchartview1.Nodes;
   var_Nodes.Add("L1_A",0,"LA",null,null).ShowLinkDir = false;
   var_Nodes.Add("L1_B",0,"LB",null,null);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
   var_Nodes.Add("L2_B","LB","LB2",null,null);
```

C# for /COM

```
axChartView1.IndentSiblingY = 30;
axChartView1.ShowLinksDir = true;
axChartView1.PenWidthLink = 2;
axChartView1.LinkColor = Color.FromArgb(0,0,0);
axChartView1.AntiAliasing = true;
EXORGCHARTLib.Nodes var_Nodes = axChartView1.Nodes;
   var_Nodes.Add("L1_A",0,"LA",null,null).ShowLinkDir = false;
   var_Nodes.Add("L1_B",0,"LB",null,null);
   var_Nodes.Add("L2_A","LA","LA2",null,null);
   var_Nodes.Add("L2_B","LB","LB2",null,null);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Node,com_Nodes;
   anytype var_Node,var_Nodes;
   ;

   super();

   exchartview1.IndentSiblingY(30);
   exchartview1.ShowLinksDir(true);
   exchartview1.PenWidthLink(2);
   exchartview1.LinkColor(WinApi::RGB2int(0,0,0));
   exchartview1.AntiAliasing(true);
   var_Nodes = exchartview1.Nodes(); com_Nodes = var_Nodes;
```

```
        var_Node =
COM::createFromObject(com_Nodes.Add("L1_A",COMVariant::createFromInt(0),"LA"));
 com_Node = var_Node;
        com_Node.ShowLinkDir(false);
        com_Nodes.Add("L1_B",COMVariant::createFromInt(0),"LB");
        com_Nodes.Add("L2_A","LA","LA2");
        com_Nodes.Add("L2_B","LB","LB2");
}
```

Delphi 8 (.NET only)

```
with AxChartView1 do
begin
   IndentSiblingY := 30;
   ShowLinksDir := True;
   PenWidthLink := 2;
   LinkColor := Color.FromArgb(0,0,0);
   AntiAliasing := True;
   with Nodes do
   begin
      Add('L1_A',TObject(0),'LA',Nil,Nil).ShowLinkDir := False;
      Add('L1_B',TObject(0),'LB',Nil,Nil);
      Add('L2_A','LA','LA2',Nil,Nil);
      Add('L2_B','LB','LB2',Nil,Nil);
   end;
end
```

Delphi (standard)

```
with ChartView1 do
begin
   IndentSiblingY := 30;
   ShowLinksDir := True;
   PenWidthLink := 2;
   LinkColor := RGB(0,0,0);
   AntiAliasing := True;
   with Nodes do
   begin
```

```
      Add('L1_A',OleVariant(0),'LA',Null,Null).ShowLinkDir := False;
      Add('L1_B',OleVariant(0),'LB',Null,Null);
      Add('L2_A','LA','LA2',Null,Null);
      Add('L2_B','LB','LB2',Null,Null);
   end;
end
```

VFP

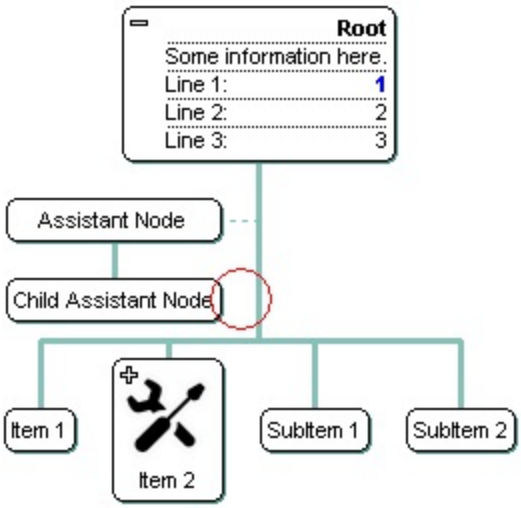
```
with thisform.ChartView1
   .IndentSiblingY = 30
   .ShowLinksDir = .T.
   .PenWidthLink = 2
   .LinkColor = RGB(0,0,0)
   .AntiAliasing = .T.
   with .Nodes
      .Add("L1_A",0,"LA").ShowLinkDir = .F.
      .Add("L1_B",0,"LB")
      .Add("L2_A","LA","LA2")
      .Add("L2_B","LB","LB2")
   endwith
endwith
```

property Node.ShowLinks as Boolean

Shows or hides the links between node and its child nodes or its parent, if is it an assistant node.

Type	Description
Boolean	A Boolean expression that indicates whether the child nodes, or assistant nodes are connected to their parent node.

By default, the ShowLinks property is True. Use the ShowLinks property to hide the connection between a child node or an assistant node to its parent. Use the LinkTo property to connect manually a node with other. The ShowLinks property doesn't affect the links added manually using the LinkTo property. Use the FixedWidth property to fix the node's width. For instance, use the ShowLinks property to hide the connection between an assistant node and its parent, so it can simulate that an assistant node can have a child assistant node, like shown in the following screen shot. The ShowLinkDir property indicates whether the direction of the link is hidden, while the ShowLinksDir property is True.



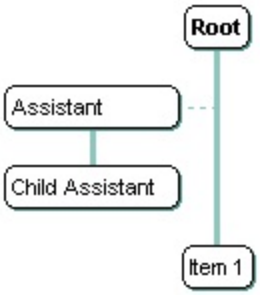
The red circle marks the area where the link between the assistant node and its parent is hidden.

The following template shows how to simulate a child assistant node:

```
BeginUpdate
PenWidthLink = 3
Root
{
   AddAssistant("Assistant")
   {
      Key = "A"
```

```
      FixedWidth = 80
   }
   AddAssistant("Child Assistant")
   {
      Left = True
      ShowLinks = False
      FixedWidth = 80
      LinkTo = "A"
   }
}
Nodes
{
   Add("Item 1")
}
EndUpdate
```

and it generates a screen like follows:



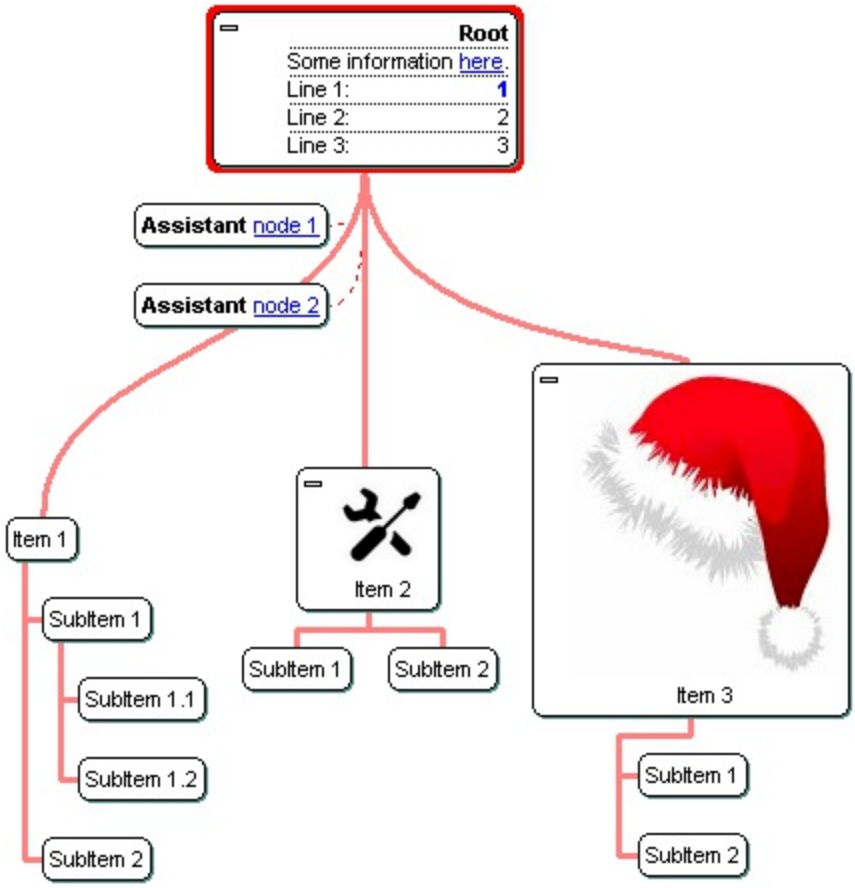
property Node.ShowRoundLink as Boolean

Specifies whether the round links are shown between parent and child nodes.

Type	Description
Boolean	A Boolean expression that specifies whether links between the node and its child nodes are shown rounded or rectangular.

By default, the ShowRoundLink property is False. Use the ShowRoundLink property to specify round links for a specified node and its child nodes. Use the ShowRoundLink property to specify round links for all nodes. Use the PenLink property indicates the type of the pen used to paint the links between nodes. Use the PenWidthLink property to specify the thickness of the links between nodes. Use the LinkColor property to specify the color for the links between nodes.

The following screen shot shows the chart using round links ONLY for root node (Root.ShowRoundLink property is True) :

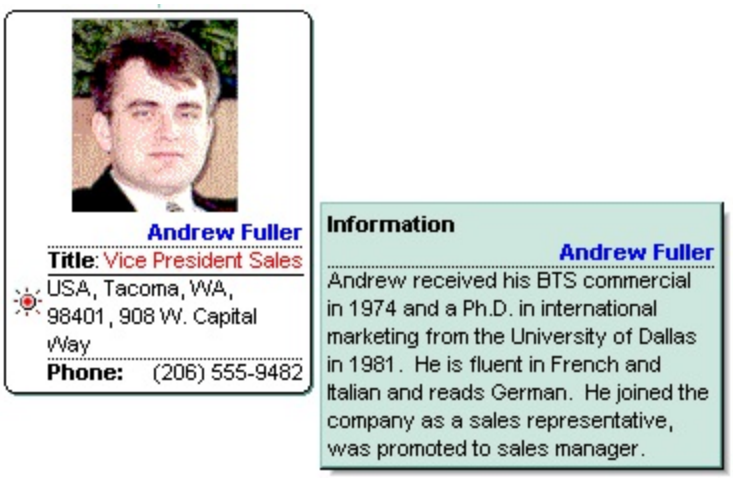


property Node.ToolTip as String

Specifies the description for the node's tooltip.

Type	Description
String	A string expression that identifies the description of the node's tooltip.

Use the ToolTip property to assign a tooltip to a node. Use the ToolTipTitle property to specify the title of the tooltip window. The tooltip shows up when the cursor hovers the node. The ToolTip property supports built-in HTML tags like the Caption property does. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the Root property to get the root node. Use the **<img>** HTML tag to insert icons inside the node's caption. 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.



The ToolTip property supports built-in HTML format like listed:

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

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

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=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

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

gradient-center

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

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

outlined

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

shadow

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

outline anti-aliasing

The following VB sample adds a node with a tooltip:

```
With ChartView1
    .BackColor = vbWhite
    With .Nodes
        With .Add("Andrew FullerTitle:Vice President Sales
USA, Tacoma, WA, 98401, 908 W. Capital Way
Phone:(206) 555-9482", "nancy", , 1, "c:\temp\sample\andrew.gif")
            .Position = 0
            .BackColor = vbWhite
            .ToolTip = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from
the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He
joined the company as a sales representative, was promoted to sales manager."
            .ToolTipTitle = "Information"
        End With
    End With
End With
```

The following C++ sample assigns a tooltip to the root node:

```
CNode node = m_chartview.GetRoot();
node.SetToolTip( "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representative, was promoted to sales manager." );
node.SetToolTipTitle( "Information" );
```

The following VB.NET sample assigns a tooltip to the root node:

```
With AxChartView1.Root
   .ToolTip = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representative, was promoted to sales manager."
   .ToolTipTitle = "Information"
End With
```

The following C# sample assigns a tooltip to the root node:

```
EXORGCHARTLib.Node node = axChartView1.Root;
node.ToolTip = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representative, was promoted to sales manager.";
node.ToolTipTitle = "Information";
```

The following VFP sample assigns a tooltip to the root node:

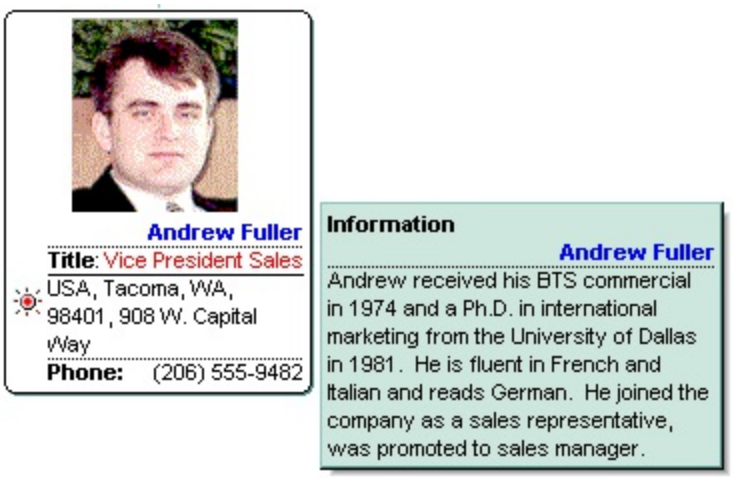
```
With thisform.ChartView1.Root
   local s
   s = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketi"
   s = s + "ng from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representati"
   .ToolTip = s + "ve, was promoted to sales manager."
   .ToolTipTitle = "Information"
EndWith
```

property Node.ToolTipTitle as String

Specifies the title of the node's tooltip.

Type	Description
String	A string expression that specifies the title of the node's tooltip.

Use the ToolTipTitle property to define the title of the tooltip of the node. Use the ToolTip property to assign a tooltip to a node. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears.



The following VB sample adds a node with a tooltip:

```
With ChartView1
    .BackColor = vbWhite
    With .Nodes
      With .Add("Andrew FullerTitle:Vice President Sales
USA, Tacoma, WA, 98401, 908 W. Capital Way
Phone:(206) 555-9482", "nancy", , 1, "c:\temp\sample\andrew.gif")
          .Position = 0
          .BackColor = vbWhite
          .ToolTip = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representative, was promoted to sales manager."
          .ToolTipTitle = "Information"
      End With
```

```
    End With
End With
```

The following C++ sample assigns a tooltip to the root node:

```
CNode node = m_chartview.GetRoot();
node.SetToolTip( "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representative, was promoted to sales manager." );
node.SetToolTipTitle( "Information" );
```

The following VB.NET sample assigns a tooltip to the root node:

```
With AxChartView1.Root
   .ToolTip = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representative, was promoted to sales manager."
   .ToolTipTitle = "Information"
End With
```

The following C# sample assigns a tooltip to the root node:

```
EXORGCHARTLib.Node node = axChartView1.Root;
node.ToolTip = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representative, was promoted to sales manager.";
node.ToolTipTitle = "Information";
```

The following VFP sample assigns a tooltip to the root node:

```
With thisform.ChartView1.Root
   local s
   s = "Andrew Fuller
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketi"
   s = s + "ng from the University of Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the company as a sales representati"
```

```
    .ToolTip = s + "ve, was promoted to sales manager."
    .ToolTipTitle = "Information"
EndWith
```

property Node.UserData as Variant

Assigns an user extra data to the node.

Type	Description
Variant	A Variant expression that identifies an extra data associated to a node.

The UserData property assigns an extra data to a node. Use the UserData property to associate a number, a string, or something else to a node. The UserData property is not used by the control. Use the Caption property to specify the caption of the node.

property Node.Width as Long

Specifies the maximum width of the node's caption.

Type	Description
Long	A long expression that defines the maximum width of the caption of the node.

Use the Width property to specify the maximum of the width of the node's caption. Use the Caption property to specify the node's caption. Use the FixedWidth and FixedHeight property to specify a fixed size for a node. Use the FixedWidthHeight and FixedHeightHeight properties to specify fixed size for all nodes in the organigram. Use the Font property to specify the control's font.

Nodes object

The Nodes collection holds the collection of nodes of the organigram. Use the Nodes property to access the Nodes collection.

Name	Description
Add	Adds a child node 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 node of the Nodes collection.
Remove	Removes a specific member from the Nodes collection.

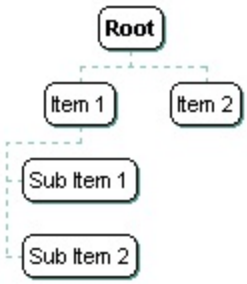
method Nodes.Add (Caption as String, [Parent as Variant], [Key as Variant], [Image as Variant], [Picture as Variant])

Adds a child node and returns a reference to the newly created object.

Type	Description
Caption as String	A string expression that identifies the node's caption. The Caption supports built-in HTML tags.
Parent as Variant	A string expression that specifies the key of the parent node, a long expression that indicates the index of the parent node, a Node object that specifies the parent node
Key as Variant	A string expression that defines the key of the node.
Image as Variant	A long expression that indicates the index of the icon being assigned to the node.
Picture as Variant	A string expression that indicates the path to the picture file being loaded by the node, a IPictureDisp object that specifies the picture of the node

Return	Description
Node	A Node object being created.

Use the Add method to add new nodes to the chart. Use the Remove method to remove a specific node. Use the Item property to find a node in the Nodes collection. By default, the Nodes collection includes the Root object. Use the BeginUpdate and EndUpdate methods to maintain performance while adding new nodes. Use the AddAssistant method to add an assistant node. An assistant node is displayed between child node and its parent. The Node being returned by the Add method has the IsAssistant property on False. Use the Image property to assign an icon to a node. Use the Picture property to assign a custom size picture to a node. Use the UserData property to associate an extra data to a node. Use the LoadXML/SaveXML methods to load/save the control's data from/to XML documents. Use the LinkCaption property to specify a caption on the line that links the parent with the current node.



The Caption parameter supports the following built-in HTML tags:

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

  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=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of

the color in hexa values.

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

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

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

Using ADO, you can load a hierarchical table, if you have a table that contains a field that identifies the key of the record ("EmployeeID"), and a field that holds the key to the parent record ("ReportsTo"), like in the following VB sample:

```
Option Explicit

Private Sub Form_Load()
On Error GoTo error
   Dim i As Long
   Dim rs As Object, t As IPictureDisp
   Set rs = CreateObject("ADODB.Recordset")
```

```
   rs.Open "Select * From Employees", "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= "
& App.Path & "\sample.mdb", 3

   With ChartView1
      .BeginUpdate
      While Not rs.EOF()
         With .Nodes
            With .Add(getInfo(rs), rs("ReportsTo").Value, rs("EmployeeID").Value)
               .ToolTip = rs("Notes") + "<br><upline><dotline><r><b>" + rs("FirstName")
+ " " + rs("LastName") + "<b>"
            End With
         End With
         rs.MoveNext
      Wend
      .EndUpdate
   End With
   Exit Sub
error:
   Label1 = "The ""ADODB.Recordset"" object is missing. Please make sure that you have
installed properly the ADO files."
End Sub

Private Function getInfo(ByVal r As Object) As String
   getInfo = r("TitleOfCourtesy") + " <b><fgcolor=000080>" + r("FirstName") + " " +
r("LastName") + "</fgcolor></b><br>" + _
   r("Title") + "<br><upline><dotline>" + _
   r("City") + "<br> " + r("Address")

End Function

Private Sub Form_Resize()
   On Error Resume Next
   With ChartView1
      .Width = Me.ScaleWidth - 2 * .Left
      .Height = Me.ScaleHeight - (Label1.Height + 3 * Label1.Top)
   End With
End Sub
```

The following VB sample adds two nodes to the Nodes collection:

```
With ChartView1
    .BeginUpdate
    .BackColor = vbWhite
      With .Nodes
          With .Add("<r><dotline><b><fgcolor=0000FF>Andrew Fuller</fgcolor></b>
<br><solidline><b>Title</b>:<r><fgcolor=FF0000>Vice President Sales</fgcolor>
<br>USA, Tacoma, WA, 98401, 908 W. Capital Way<br><dotline><upline><b>Phone:
</b><r>(206) 555-9482", , "andrewKey", 1, "c:\temp\sample\andrew.gif")
              .Position = 0
              .BackColor = vbWhite
              .ToolTip = "<dotline><r><b><fgcolor=0000FF>Andrew Fuller</fgcolor>
</b><br>Andrew received his BTS commercial in 1974 and a Ph.D. in international
marketing from the University of Dallas in 1981.  He is fluent in French and Italian and
reads German.  He joined the company as a sales representative, was promoted to sales
manager."
              .ToolTipTitle = "Information"
          End With
          With .Add("<r><dotline><b><fgcolor=0000FF>Steven Buchanan</fgcolor>
</b><br><solidline><b>Title</b>:<r>Sales Manager<br>UK, London, SW1 8JR, 14
Garrett Hill<br><dotline><upline><b>Phone:</b><r>(71) 555-4848<br><dotline>
<upline>Hire Date:<r>17-Oct-1993", "andrewKey", "stevenKey", ,
"c:\temp\sample\steven.gif")
          End With
      End With
    .EndUpdate
End With
```

The following VB sample adds few nodes to the control like shown above:

```
With ChartView1
  .BeginUpdate
  With .Nodes
    .Add "Item 1", "root", "Key1"
    .Add "Item 2", "root"
    .Add "Sub Item 1", "Key1"
    .Add "Sub Item 2", "Key1"
```

```
    End With
    .EndUpdate
End With
```

The following C++ sample adds few nodes to the control like shown above:

```
#include "nodes.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
m_chartview.BeginUpdate();
CNodes nodes = m_chartview.GetNodes();
nodes.Add( "Item 1", COleVariant( "root" ), COleVariant("Key1"), vtMissing, vtMissing );
nodes.Add( "Item 2", COleVariant( "root" ), vtMissing, vtMissing, vtMissing );
nodes.Add( "Sub Item 1", COleVariant( "Key1" ), vtMissing, vtMissing, vtMissing );
nodes.Add( "Sub Item 2", COleVariant( "Key1" ), vtMissing, vtMissing, vtMissing );
m_chartview.EndUpdate();
```

The following VB.NET sample adds few nodes to the control like shown above:

```
With AxChartView1
    .BeginUpdate()
    With .Nodes
        .Add("Item 1", "root", "Key1")
        .Add("Item 2", "root")
        .Add("Sub Item 1", "Key1")
        .Add("Sub Item 2", "Key1")
    End With
    .EndUpdate()
End With
```

The following C# sample adds few nodes to the control like shown above:

```
axChartView1.BeginUpdate();
EXORGCHARTLib.Nodes nodes = axChartView1.Nodes;
nodes.Add("Item 1", "root", "Key1", null, null);
nodes.Add("Item 2", "root", null, null, null);
nodes.Add("Sub Item 1", "Key1", null, null, null);
nodes.Add("Sub Item 2", "Key1", null, null, null);
axChartView1.EndUpdate();
```

The following VFP sample adds few nodes to the control like shown above:

```
With thisform.ChartView1
   .BeginUpdate
   With .Nodes
      .Add("Item 1", "root", "Key1")
      .Add("Item 2", "root")
      .Add("Sub Item 1", "Key1")
      .Add("Sub Item 2", "Key1")
   EndWith
   .EndUpdate
EndWith
```

method Nodes.Clear ()

Removes all objects in a collection.

Type	Description

Use the Clear method to clear the entire nodes collection. Use the Remove method to remove a specific node in the collection. The ClearAssistants method clears the assistant nodes collection. Use the ShowAssistants property to hide all assistant nodes. Use the Expanded property to expand or collapse a node. Use the Count property to count the nodes in the collection. Use the Item property to access a node giving its index or its key. Use the Index property to get the index of the node. Use the Key property to specify the key of the node.

property Nodes.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A long expression that specifies the number of nodes in the organigram

Use the Count property to get the number of nodes in the Nodes collection. Use the Item property to access the node by key or by its index. Use the Index property to get the index of the node. Use the Key property to specify the key of the node. Use the FirstNode property to retrieve the first child node. Use the NextNode property to determine the next sibling node. Use the NodeCount property to get the number of child nodes. Use the Position property to change the node's position.

The following VB sample enumerates the nodes in the control:

```
Dim i As Long
With ChartView1.Nodes
   For i = 0 To .Count - 1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .Item(i)
      Debug.Print n.Caption
   Next
End With
```

The following VB sample enumerates the nodes in the control:

```
Dim n As EXORGCHARTLibCtl.Node
For Each n In ChartView1.Nodes
   Debug.Print n.Caption
Next
```

The following C++ sample enumerates the nodes in the control:

```
#include "nodes.h"
CNodes nodes = m_chartview.GetNodes();
for ( long i = 0; i < nodes.GetCount(); i++ )
{
   CNode node = nodes.GetItem( COleVariant( i ) );
```

```
    OutputDebugString( node.GetCaption() );
}
```

The following VB.NET sample enumerates the nodes in the control:

```
Dim i As Integer
With AxChartView1.Nodes
    For i = 0 To .Count - 1
        Dim n As EXORGCHARTLib.Node = .Item(i)
        Debug.WriteLine(n.Caption)
    Next
End With
```

The following VB.NET sample enumerates the nodes in the control:

```
Dim n As EXORGCHARTLib.Node
For Each n In AxChartView1.Nodes
    Debug.WriteLine(n.Caption)
Next
```

The following C# sample enumerates the nodes in the control:

```
for (int i = 0; i < axChartView1.Nodes.Count; i++)
    System.Diagnostics.Debug.WriteLine(axChartView1.Nodes[i].Caption);
```

The following VFP sample enumerates the nodes in the control:

```
local i
With thisform.ChartView1.Nodes
    For i = 0 To .Count - 1
        local n
        n = .Item(i)
        wait window nowait n.Caption
    Next
EndWith
```

property Nodes.Item (Index as Variant) as Node

Returns a specific node of the Nodes collection.

Type	Description
Index as Variant	A string expression that specifies the key of the node, a long expression that indicates the index of the node, a Node object that specifies the node being accessed.
Node	A Node object being accessed.

Use the Item property to access the node by key or by its index. Use the Count property to count the nodes in the collection. Use the Index property to get the index of the node. Use the Key property to specify the key of the node. Use the FirstNode property to retrieve the first child node. Use the NextNode property to determine the next sibling node. Use the NodeCount property to get the number of child nodes. Use the Position property to change the node's position.

The following VB sample enumerates the nodes in the control:

```
Dim i As Long
With ChartView1.Nodes
   For i = 0 To .Count - 1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .Item(i)
      Debug.Print n.Caption
   Next
End With
```

The following VB sample enumerates the nodes in the control:

```
Dim n As EXORGCHARTLibCtl.Node
For Each n In ChartView1.Nodes
   Debug.Print n.Caption
Next
```

The following C++ sample enumerates the nodes in the control:

```
#include "nodes.h"
CNodes nodes = m_chartview.GetNodes();
for ( long i = 0; i < nodes.GetCount(); i++ )
```

```
{
   CNode node = nodes.GetItem( COleVariant( i ) );
   OutputDebugString( node.GetCaption() );
}
```

The following VB.NET sample enumerates the nodes in the control:

```
Dim i As Integer
With AxChartView1.Nodes
   For i = 0 To .Count - 1
      Dim n As EXORGCHARTLib.Node = .Item(i)
      Debug.WriteLine(n.Caption)
   Next
End With
```

The following VB.NET sample enumerates the nodes in the control:

```
Dim n As EXORGCHARTLib.Node
For Each n In AxChartView1.Nodes
   Debug.WriteLine(n.Caption)
Next
```

The following C# sample enumerates the nodes in the control:

```
for (int i = 0; i < axChartView1.Nodes.Count; i++)
   System.Diagnostics.Debug.WriteLine(axChartView1.Nodes[i].Caption);
```

The following VFP sample enumerates the nodes in the control:

```
local i
With thisform.ChartView1.Nodes
   For i = 0 To .Count - 1
      local n
      n = .Item(i)
      wait window nowait n.Caption
   Next
EndWith
```

method Nodes.Remove (Index as Variant)

Removes a specific member from the Nodes collection.

Type	Description
Index as Variant	A string expression that specifies the key of the node, a long expression that indicates the index of the node, a Node object that specifies the node being removed.

Use the Remove method to remove a child node. Use the Clear method to clear all nodes in the collection. The Root node (Index = 0) can't be removed. Use the RemoveAssistant method to remove an assistant node. The IsAssistant property specifies whether a node is a child node or an assistant node. Use the ShowAssistants property to hide all assistant nodes. Use the Expanded property to collapse a node. The Remove method does not remove recursively the child nodes. *Use the Remove method of the Node object to remove recursively the child nodes, the assistant nodes and all the nodes in the same group, and if possible remove the node itself.*

The following VB sample removes recursively the node all all its child nodes:

```
Private Sub removeRec(ByVal chart As EXORGCHARTLibCtl.ChartView, ByVal n As
EXORGCHARTLibCtl.Node)
   Dim c As EXORGCHARTLibCtl.Node
   Set c = n.FirstNode
   While Not (c Is Nothing)
      Dim x As EXORGCHARTLibCtl.Node
      Set x = c.NextNode
      removeRec chart, c
      Set c = x
   Wend
   If Not (n.Parent Is Nothing) Then
      chart.Nodes.Remove n
   End If
End Sub
```

The following C++ sample removes recursively the node all all its child nodes:

```
void removeRec( CChartView* pChart, CNode* pNode )
{
   if ( pNode != NULL )
```

```
    {
        CNode child = pNode->GetFirstNode();
        while ( child.m_lpDispatch != NULL )
        {
            CNode next = child.GetNextNode();
            removeRec( pChart, &child );
            child = next;
        }
        if ( pNode->GetParent().m_lpDispatch != NULL )
            pChart->GetNodes().Remove( COleVariant( pNode->GetKey() ) );
    }
}
```

and you can call the removeRec function like follows:

```
CNode node( V_DISPATCH( &m_chartview.GetSelectNode() ) );
removeRec( &m_chartview, &node );
```

The following VB.NET sample removes recursively the node all all its child nodes:

```
Private Sub removeRec(ByVal chart As AxEXORGCHARTLib.AxChartView, ByVal n As
EXORGCHARTLib.Node)
    Dim c As EXORGCHARTLib.Node = n.FirstNode
    While Not (c Is Nothing)
        Dim x As EXORGCHARTLib.Node = c.NextNode
        removeRec(chart, c)
        c = x
    End While
    If Not (n.Parent Is Nothing) Then
        chart.Nodes.Remove(n)
    End If
End Sub
```

The following C# sample removes recursively the node all all its child nodes:

```
private void removeRec(AxEXORGCHARTLib.AxChartView chart, EXORGCHARTLib.Node n)
{
    EXORGCHARTLib.Node c = n.FirstNode;
    while (c != null)
```

```
    {
        EXORGCHARTLib.Node x = c.NextNode;
        removeRec(chart,c);
        c = x;
    }
    if ( n.Parent != null )
        chart.Nodes.Remove(n);
}
```

and you can call the removeRec function like follows:

```
removeRec(axChartView1, axChartView1.SelectNode as EXORGCHARTLib.Node);
```

The following VFP sample removes recursively the node all all its child nodes (add removerec method):

```
lparameters chart, n

local c
c = n.FirstNode
do While !(isnull(c))
    local x
    with c
        x = .NextNode
    endwith
    thisform.removerec(chart, c)
    c = x
enddo
If !(isnull(n.Parent))
    chart.Nodes.Remove(n)
EndIf
```

Pattern object

The Pattern object can be used to apply a pattern and a frame with different colors on an UI element. For instance, the Pattern property indicates the pattern to be applied on the frame. The Pattern object supports the following properties:

Name	Description
Color	Specifies the pattern color.
FrameColor	Specifies the pattern's frame color.
Type	Retrieves or sets a value that indicates the pattern to fill the element.

property Pattern.Color as Color

Specifies the pattern color.

Type	Description
Color	A Color expression that specifies the color to show the pattern.

By default, the Color property is 0 (black). The Color property indicates the color to display the pattern. The Type property indicates the type of the pattern to be shown. The FrameColor property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property.

property Pattern.FrameColor as Color

Specifies the pattern's frame color.

Type	Description
Color	A Color expression that specifies the color to show the frame.

By default, the FrameColor property is 0 (black). The FrameColor property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The Type property indicates the type of the pattern to be shown. The Color property indicates the color to display the pattern.

property Pattern.Type as PatternEnum

Retrieves or sets a value that indicates the pattern to fill the element.

Type	Description
PatternEnum	A PatternEnum expression that specifies the type of the pattern to fill the element.

By default, the Type property is exPatternFrameThick which indicates that a thick-border is shown. The Type property indicates the pattern to display on the element. The Color property indicates the color to display the pattern. The FrameColor property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property.

ExOrgChart events

The ExOrgChart component supports the following events:

Name	Description
AddNew	Occurs when the user clicks any of the add new buttons.
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user presses and then releases the left mouse button over the control.
DblClick	Occurs when the user dblclk the left mouse button over an object.
DropFile	Notifies whether the user drags a file over a node.
Event	Notifies the application once the control fires an event.
Expand	Occurs when the user expands or collapses a node.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
LayoutEndChanging	Notifies your application once the control's layout has been changed.
LayoutStartChanging	Occurs when the control's layout is about to be changed.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
ScrollButtonClick	Occurs when the user clicks a button in the scrollbar.
Select	Occurs when the user selects a node.
ToolTip	Fired when the control prepares the object's tooltip.

event AddNew (AddNewType as Long)

Occurs when the user clicks any of the add new buttons.

Type	Description
AddNewType as Long	A long expression that indicates the type of the button being clicked.

The AddNew event notifies your application once the user clicks an add new button. Use the AddNew event to add new parent, child or assistant node once the user clicks an add new button.

Syntax for AddNew event, **/NET** version, on:

C#
```
private void AddNew(object sender,int AddNewType)
{
}
```

VB
```
Private Sub AddNew(ByVal sender As System.Object,ByVal AddNewType As
Integer) Handles AddNew
End Sub
```

Syntax for AddNew event, **/COM** version, on:

C#
```
private void AddNew(object sender,
AxEXORGCHARTLib._IChartViewEvents_AddNewEvent e)
{
}
```

C++
```
void OnAddNew(long AddNewType)
{
}
```

C++ Builder
```
void __fastcall AddNew(TObject *Sender,long AddNewType)
{
}
```

Delphi
```
procedure AddNew(ASender: TObject; AddNewType : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AddNew(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_AddNewEvent);
begin
end;
```

Powe…

```
begin event AddNew(long AddNewType)
end event AddNew
```

VB.NET

```
Private Sub AddNew(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_AddNewEvent) Handles AddNew
End Sub
```

VB6

```
Private Sub AddNew(ByVal AddNewType As Long)
End Sub
```

VBA

```
Private Sub AddNew(ByVal AddNewType As Long)
End Sub
```

VFP

```
LPARAMETERS AddNewType
```

Xbas…

```
PROCEDURE OnAddNew(oChartView,AddNewType)
RETURN
```

Syntax for AddNew event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AddNew(AddNewType)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AddNew(AddNewType)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAddNew Integer llAddNewType
    Forward Send OnComAddNew llAddNewType
End_Procedure
```

Visual Objects

```
METHOD OCX_AddNew(AddNewType) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AddNew(int _AddNewType)
{
}
```

XBasic

```
function AddNew as v (AddNewType as N)
end function
```

dBASE

```
function nativeObject_AddNew(AddNewType)
return
```

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Type	Description
AnchorID as String	A string expression that indicates the identifier of the anchor
Options as String	A string expression that specifies options of the anchor element.

The control fires the AnchorClick event to notify that the user clicks an anchor element. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The AnchorClick event is fired only if prior clicking the control it shows the hand cursor. For instance, if the cell is disabled, the hand cursor is not shown when hovers the anchor element, and so the AnchorClick event is not fired. Use the FormatAnchor property to specify the visual effect for anchor elements. For instance, if the user clicks the anchor *<a1>anchor*, the control fires the AnchorClick event, where the AnchorID parameter is 1, and the Options parameter is empty. Also, if the user clicks the anchor *<a 1;yourextradata>anchor*, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "yourextradata".

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,
AxEXORGCHARTLib._IChartViewEvents_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:
AxEXORGCHARTLib._IChartViewEvents_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
AxEXORGCHARTLib._IChartViewEvents_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(oChartView,AnchorID,Options)
RETURN
```

Syntax for AnchorClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AnchorClick(AnchorID,Options)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AnchorClick(AnchorID,Options)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAnchorClick String llAnchorID String llOptions
    Forward Send OnComAnchorClick llAnchorID llOptions
End_Procedure
```

Visual Objects

```
METHOD OCX_AnchorClick(AnchorID,Options) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AnchorClick(str _AnchorID,str _Options)
{
}
```

XBasic

```
function AnchorClick as v (AnchorID as C,Options as C)
end function
```

dBASE

```
function nativeObject_AnchorClick(AnchorID,Options)
return
```

event Click ()

Occurs when the user presses and then releases the left mouse button over the control.

Type	Description

The Click event is fired when the user releases the left mouse button over the control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. The control fires the Select event when the user clicks a node (selects a node). Use the SelectNode property to specify the selected node. The ScrollByClick property specifies a value that indicates whether the user scrolls the control's content by clicking the client area.

Syntax for Click event, **/NET** version, on:

C#

```
private void Click(object sender)
{
}
```

VB

```
Private Sub Click(ByVal sender As System.Object) Handles Click
End Sub
```

Syntax for Click event, **/COM** version, on:

C#

```
private void ClickEvent(object sender, EventArgs e)
{
}
```

C++

```
void OnClick()
{
}
```

C++ Builder

```
void __fastcall Click(TObject *Sender)
{
}
```

Delphi

```
procedure Click(ASender: TObject; );
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure ClickEvent(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event Click()
end event Click
```

VB.NET

```
Private Sub ClickEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ClickEvent
End Sub
```

VB6

```
Private Sub Click()
End Sub
```

VBA

```
Private Sub Click()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnClick(oChartView)
RETURN
```

Syntax for Click event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Click()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Click()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComClick
   Forward Send OnComClick
End_Procedure
```

Visual Objects

```
METHOD OCX_Click() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Click()
{
}
```

XBasic

```
function Click as v ()
end function
```

dBASE

```
function nativeObject_Click()
return
```

event DblClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user 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.

Use the DblClick event to notify your application that user double clicks a node. Use the NodeFromPoint property to determine the node from the point. Use the ExpandOnDblClick property to specify whether a node is expanded or collapsed when the user double clicks it. Use the Expanded property to expand or collapse a node. Use the Caption property to specify the node of the caption.

Syntax for DblClick event, **/NET** version, on:

C#

```
private void DblClick(object sender,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub DblClick(ByVal sender As System.Object,ByVal Shift As Short,ByVal X
As Integer,ByVal Y As Integer) Handles DblClick
End Sub
```

Syntax for DblClick event, **/COM** version, on:

C#

```
private void DblClick(object sender,
AxEXORGCHARTLib._IChartViewEvents_DblClickEvent e)
{
}
```

C++

```
void OnDblClick(short Shift,long X,long Y)
{
```

```
}
```

C++ Builder

```
void __fastcall DblClick(TObject *Sender,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure DblClick(ASender: TObject; Shift : Smallint;X : Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DblClick(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_DblClickEvent);
begin
end;
```

Powe…

```
begin event DblClick(integer Shift,long X,long Y)
end event DblClick
```

VB.NET

```
Private Sub DblClick(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_DblClickEvent) Handles DblClick
End Sub
```

VB6

```
Private Sub DblClick(Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub DblClick(ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Shift,X,Y
```

Xbas…

```
PROCEDURE OnDblClick(oChartView,Shift,X,Y)
RETURN
```

Syntax for DblClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="DblClick(Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DblClick(Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDblClick Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
   Forward Send OnComDblClick llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_DblClick(Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DblClick(int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function DblClick as v (Shift as N,X as
OLE::Exontrol.ChartView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ChartView.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_DblClick(Shift,X,Y)
return
```

The following VB sample determines the node that user double clicks:

```
Private Sub ChartView1_DblClick(Shift As Integer, X As Single, Y As Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .NodeFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (n Is Nothing) Then
         MsgBox n.Caption
      End If
   End With
End Sub
```

The following C++ sample determines the node being double clicked:

```
#include "node.h"
void OnDblClickChartview1(short Shift, long X, long Y)
{
   CNode node = m_chartview.GetNodeFromPoint( X, Y );
   if ( node.m_lpDispatch != NULL )
      MessageBox( node.GetCaption() );
}
```

The following VB.NET sample determines the node being double clicked:

```
Private Sub AxChartView1_DblClick(ByVal sender As Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_DblClickEvent) Handles AxChartView1.DblClick
   With AxChartView1
      Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(e.x, e.y)
      If Not (n Is Nothing) Then
         MsgBox(n.Caption)
      End If
   End With
End Sub
```

The following C# sample determines the node being double clicked:

```
private void axChartView1_DblClick(object sender,
AxEXORGCHARTLib._IChartViewEvents_DblClickEvent e)
{
   EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(e.x, e.y);
   if (node != null)
      MessageBox.Show(node.Caption);
}
```

The following VFP sample determines the node being double clicked:

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

With thisform.ChartView1
   local n
   n = .NodeFromPoint(x , y )
```

```
    If !isnull(n) Then
        wait window nowait n.Caption
    EndIf
EndWith
```

event DropFile (File as String, Node as Node)

Notifies whether the user drags a file over a node.

Type	Description
File as String	A String expression that specifies the name of the file being dragged.
Node as Node	A Node object that indicates the node where the user drags the file.

Use the DropFile event to notifiy your application when the user drags a file over a node. Use the AcceptFiles property to specify whether the control supports files by drag and drop. The DropFile event is not filed if the AcceptFiles property is False.

Syntax for DropFile event, **/NET** version, on:

C#

```
private void DropFile(object sender,string File,exontrol.EXORGCHARTLib.Node Node)
{
}
```

VB

```
Private Sub DropFile(ByVal sender As System.Object,ByVal File As String,ByVal Node As exontrol.EXORGCHARTLib.Node) Handles DropFile
End Sub
```

Syntax for DropFile event, **/COM** version, on:

C#

```
private void DropFile(object sender, AxEXORGCHARTLib._IChartViewEvents_DropFileEvent e)
{
}
```

C++

```
void OnDropFile(LPCTSTR File,LPDISPATCH Node)
{
}
```

C++ Builder

```
void __fastcall DropFile(TObject *Sender,BSTR File,Exorgchartlib_tlb::INode *Node)
{
}
```

Delphi

```
procedure DropFile(ASender: TObject; File : WideString;Node : INode);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DropFile(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_DropFileEvent);
begin
end;
```

Powe…

```
begin event DropFile(string File,oleobject Node)
end event DropFile
```

VB.NET

```
Private Sub DropFile(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_DropFileEvent) Handles DropFile
End Sub
```

VB6

```
Private Sub DropFile(ByVal File As String,ByVal Node As
EXORGCHARTLibCtl.INode)
End Sub
```

VBA

```
Private Sub DropFile(ByVal File As String,ByVal Node As Object)
End Sub
```

VFP

```
LPARAMETERS File,Node
```

Xbas…

```
PROCEDURE OnDropFile(oChartView,File,Node)
RETURN
```

Syntax for DropFile event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="DropFile(File,Node)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DropFile(File,Node)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComDropFile String llFile Variant llNode
    Forward Send OnComDropFile llFile llNode
End_Procedure
```

Visual Objects

```
METHOD OCX_DropFile(File,Node) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DropFile(str _File,COM _Node)
{
}
```

XBasic

```
function DropFile as v (File as C,Node as OLE::Exontrol.ChartView.1::INode)
end function
```

dBASE

```
function nativeObject_DropFile(File,Node)
return
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

Type	Description
EventID as Long	A Long expression that specifies the identifier of the event. Each internal event of the control has an unique identifier. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties). The EventParam(-1) retrieves the number of parameters of fired event

The Event notification occurs ANY time the control fires an event. *This is useful for X++, which does not support event with parameters passed by reference. Also, this could be useful for C++ Builder or Delphi, which does not handle properly the events with parameters of VARIANT type.*

In X++ the "Error executing code: FormActiveXControl (data source), method ... called with invalid parameters" occurs when handling events that have parameters passed by reference. Passed by reference, means that in the event handler, you can change the value for that parameter, and so the control will takes the new value, and use it. The X++ is NOT able to handle properly events with parameters by reference, so we have the solution.

The solution is using and handling the Event notification and EventParam method., instead handling the event that gives the "invalid parameters" error executing code.

If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print exChartView1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

Syntax for Event event, **/NET** version, on:

C#
```
private void Event(object sender,int EventID)
{
}
```

VB
```
Private Sub Event(ByVal sender As System.Object,ByVal EventID As Integer)
Handles Event
End Sub
```

Syntax for Event event, **/COM** version, on:

C#
```
private void Event(object sender,
AxEXORGCHARTLib._IChartViewEvents_EventEvent e)
{
}
```

C++
```
void OnEvent(long EventID)
{
}
```

C++ Builder
```
void __fastcall Event(TObject *Sender,long EventID)
{
}
```

Delphi
```
procedure Event(ASender: TObject; EventID : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure Event(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_EventEvent);
begin
```

```
end;
```

Powe…

```
begin event Event(long EventID)
end event Event
```

VB.NET

```
Private Sub Event(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_EventEvent) Handles Event
End Sub
```

VB6

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VBA

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VFP

```
LPARAMETERS EventID
```

Xbas…

```
PROCEDURE OnEvent(oChartView,EventID)
RETURN
```

Syntax for Event event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Event(EventID)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Event(EventID)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComEvent Integer llEventID
   Forward Send OnComEvent llEventID
End_Procedure
```

Visual Objects

```
METHOD OCX_Event(EventID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Event(int _EventID)
{
}
```

XBasic

```
function Event as v (EventID as N)
end function
```

dBASE

```
function nativeObject_Event(EventID)
return
```

event Expand (NewNode as Node)

Occurs when the user expands or collapses a node.

Type	Description
NewNode as Node	A Node object being expanded or collapsed.

Use the Expand event to notify your application that user expands or collapses a node. Use the HasButtons property to display +/- buttons for nodes that contain child nodes. Use the HasButton property to hide the +/- button for a particular node. Use the Expanded property to programmatically expand a node. Use the SelectNode property to select a node. Use the EnsureVisibleNode method to ensure that a node fits the control's visible client area. Use the ExpandOnDblClick property to specify whether a node is expanded or collapsed when the user double clicks it.

Syntax for Expand event, **/NET** version, on:

C#

```
private void Expand(object sender,exontrol.EXORGCHARTLib.Node NewNode)
{
}
```

VB

```
Private Sub Expand(ByVal sender As System.Object,ByVal NewNode As
exontrol.EXORGCHARTLib.Node) Handles Expand
End Sub
```

Syntax for Expand event, **/COM** version, on:

C#

```
private void Expand(object sender,
AxEXORGCHARTLib._IChartViewEvents_ExpandEvent e)
{
}
```

C++

```
void OnExpand(LPDISPATCH NewNode)
{
}
```

C++ Builder

```
void __fastcall Expand(TObject *Sender,Exorgchartlib_tlb::INode *NewNode)
{
}
```

Delphi

```
procedure Expand(ASender: TObject; NewNode : INode);
```

```
begin
end;
```

Delphi 8 (.NET only)

```
procedure Expand(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_ExpandEvent);
begin
end;
```

Powe…

```
begin event Expand(oleobject NewNode)
end event Expand
```

VB.NET

```
Private Sub Expand(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_ExpandEvent) Handles Expand
End Sub
```

VB6

```
Private Sub Expand(ByVal NewNode As EXORGCHARTLibCtl.INode)
End Sub
```

VBA

```
Private Sub Expand(ByVal NewNode As Object)
End Sub
```

VFP

```
LPARAMETERS NewNode
```

Xbas…

```
PROCEDURE OnExpand(oChartView,NewNode)
RETURN
```

Syntax for Expand event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Expand(NewNode)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Expand(NewNode)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComExpand Variant llNewNode
   Forward Send OnComExpand llNewNode
End_Procedure
```

Visual Objects

```
METHOD OCX_Expand(NewNode) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Expand(COM _NewNode)
{
}
```

XBasic

```
function Expand as v (NewNode as OLE::Exontrol.ChartView.1::INode)
end function
```

dBASE

```
function nativeObject_Expand(NewNode)
return
```

The following VB sample selects and ensures that the node being expanded is visible:

```
Private Sub ChartView1_Expand(ByVal NewNode As EXORGCHARTLibCtl.INode)
   With ChartView1
      .EnsureVisibleNode NewNode
      .SelectNode = NewNode
   End With
End Sub
```

The following C++ sample displays the caption of the node being expanded or collapsed:

```
void OnExpandChartview1(LPDISPATCH NewNode)
{
   CNode node( NewNode ); node.m_bAutoRelease = FALSE;
   OutputDebugString( node.GetCaption() );
}
```

The following VB.NET sample displays the caption of the node being expanded or collapsed:

```
Private Sub AxChartView1_Expand(ByVal sender As Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_ExpandEvent) Handles AxChartView1.Expand
   Debug.WriteLine(e.newNode.Caption)
End Sub
```

The following C# sample displays the caption of the node being expanded or collapsed:

```
private void axChartView1_Expand(object sender,
AxEXORGCHARTLib._IChartViewEvents_ExpandEvent e)
{
   System.Diagnostics.Debug.WriteLine(e.newNode.Caption);
}
```

The following VFP sample displays the caption of the node being expanded or collapsed:

```
*** ActiveX Control Event ***
LPARAMETERS newnode

with newnode
   wait window nowait .Caption
endwith
```

event KeyDown (KeyCode as Integer, Shift as Integer)

Occurs when the user presses a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key. You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the SelectNode property to determine the selected node. 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,
AxEXORGCHARTLib._IChartViewEvents_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:
AxEXORGCHARTLib._IChartViewEvents_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
AxEXORGCHARTLib._IChartViewEvents_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(oChartView,KeyCode,Shift)
RETURN
```

Syntax for KeyDown event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function KeyDown(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComKeyDown Short llKeyCode Short llShift
   Forward Send OnComKeyDown llKeyCode llShift
End_Procedure
```

Visual Objects
```
METHOD OCX_KeyDown(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_KeyDown(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic
```
function KeyDown as v (KeyCode as N,Shift as N)
end function
```

dBASE
```
function nativeObject_KeyDown(KeyCode,Shift)
return
```

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	An integer that returns a standard numeric ANSI keycode.

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress does not indicate the physical state of the keyboard; instead, it passes a character. KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters. Use the SelectNode property to determine the selected node.

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,
AxEXORGCHARTLib._IChartViewEvents_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:
AxEXORGCHARTLib._IChartViewEvents_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
AxEXORGCHARTLib._IChartViewEvents_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(oChartView,KeyAscii)
RETURN
```

Syntax for KeyPress event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="KeyPress(KeyAscii)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyPress(KeyAscii)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyPress Short llKeyAscii
   Forward Send OnComKeyPress llKeyAscii
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii)
{
}
```

XBasic

```
function KeyPress as v (KeyAscii as N)
end function
```

dBASE

```
function nativeObject_KeyPress(KeyAscii)
return
```

event KeyUp (KeyCode as Integer, Shift as Integer)

Occurs when the user releases a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the KeyUp event procedure to respond to the releasing of a key. Use the SelectNode property to determine the selected node.

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,
AxEXORGCHARTLib._IChartViewEvents_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:
AxEXORGCHARTLib._IChartViewEvents_KeyUpEvent);
begin
end;
```

Powe…

```
begin event KeyUp(integer KeyCode,integer Shift)
end event KeyUp
```

VB.NET

```
Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_KeyUpEvent) Handles KeyUpEvent
End Sub
```

VB6

```
Private Sub KeyUp(KeyCode As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyUp(KeyCode As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift
```

Xbas…

```
PROCEDURE OnKeyUp(oChartView,KeyCode,Shift)
RETURN
```

Syntax for KeyUp event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="KeyUp(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyUp(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyUp Short llKeyCode Short llShift
   Forward Send OnComKeyUp llKeyCode llShift
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyUp(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyUp(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic

```
function KeyUp as v (KeyCode as N,Shift as N)
end function
```

dBASE

```
function nativeObject_KeyUp(KeyCode,Shift)
return
```

event LayoutEndChanging (Operation as LayoutChangingEnum)

Notifies your application once the control's layout has been changed.

Type	Description
Operation as LayoutChangingEnum	An LayoutChangingEnum expression that specifies whether an UI operation ends.

The LayoutEndChanging event notifies your application once of the following operation ends:

- the user edits a node
- the user resizes the chart at runtime, using the middle mouse button
- the user moves a node from one parent to another.

The LayoutStartChanging event notifies once the user starts any of these operations. You can use the NodeFromPoint property to get the node from the current position.

Syntax for LayoutEndChanging event, **/NET** version, on:

C#

```
private void LayoutEndChanging(object
sender,exontrol.EXORGCHARTLib.LayoutChangingEnum Operation)
{
}
```

VB

```
Private Sub LayoutEndChanging(ByVal sender As System.Object,ByVal Operation
As exontrol.EXORGCHARTLib.LayoutChangingEnum) Handles LayoutEndChanging
End Sub
```

Syntax for LayoutEndChanging event, **/COM** version, on:

C#

```
private void LayoutEndChanging(object sender,
AxEXORGCHARTLib._IChartViewEvents_LayoutEndChangingEvent e)
{
}
```

C++

```
void OnLayoutEndChanging(long Operation)
{
}
```

C++ Builder

```
void __fastcall LayoutEndChanging(TObject
```

```
*Sender,Exorgchartlib_tlb::LayoutChangingEnum Operation)
{
}
```

Delphi

```
procedure LayoutEndChanging(ASender: TObject; Operation :
LayoutChangingEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure LayoutEndChanging(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_LayoutEndChangingEvent);
begin
end;
```

Powe…

```
begin event LayoutEndChanging(long Operation)
end event LayoutEndChanging
```

VB.NET

```
Private Sub LayoutEndChanging(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_LayoutEndChangingEvent) Handles
LayoutEndChanging
End Sub
```

VB6

```
Private Sub LayoutEndChanging(ByVal Operation As
EXORGCHARTLibCtl.LayoutChangingEnum)
End Sub
```

VBA

```
Private Sub LayoutEndChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation
```

Xbas…

```
PROCEDURE OnLayoutEndChanging(oChartView,Operation)
RETURN
```

Syntax for LayoutEndChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="LayoutEndChanging(Operation)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function LayoutEndChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComLayoutEndChanging OLELayoutChangingEnum llOperation
    Forward Send OnComLayoutEndChanging llOperation
End_Procedure
```

Visual Objects

```
METHOD OCX_LayoutEndChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_LayoutEndChanging(int _Operation)
{
}
```

XBasic

```
function LayoutEndChanging as v (Operation as
OLE::Exontrol.ChartView.1::LayoutChangingEnum)
end function
```

dBASE

```
function nativeObject_LayoutEndChanging(Operation)
return
```

event LayoutStartChanging (Operation as LayoutChangingEnum)

Occurs when the control's layout is about to be changed.

Type	Description
Operation as LayoutChangingEnum	An LayoutChangingEnum expression that specifies whether an UI operation is started.

The LayoutStartChanging event notifies your application once of the following operation starts:

- the user edits a node
- the user resizes the chart at runtime, using the middle mouse button
- the user moves a node from one parent to another.

The LayoutEndChanging event notifies once the user done any of these operations. You can use the NodeFromPoint property to get the node from the current position. The Background(exEditNodeBackColor)/Background(exEditNodeForeColor) property specifies the background/foreground color of the edit field being displayed on the node while editing.

Syntax for LayoutStartChanging event, **/NET** version, on:

C#

```
private void LayoutStartChanging(object
sender,exontrol.EXORGCHARTLib.LayoutChangingEnum Operation)
{
}
```

VB

```
Private Sub LayoutStartChanging(ByVal sender As System.Object,ByVal Operation
As exontrol.EXORGCHARTLib.LayoutChangingEnum) Handles
LayoutStartChanging
End Sub
```

Syntax for LayoutStartChanging event, **/COM** version, on:

C#

```
private void LayoutStartChanging(object sender,
AxEXORGCHARTLib._IChartViewEvents_LayoutStartChangingEvent e)
{
}
```

C++

```
void OnLayoutStartChanging(long Operation)
{
}
```

C++ Builder

```
void __fastcall LayoutStartChanging(TObject
*Sender,Exorgchartlib_tlb::LayoutChangingEnum Operation)
{
}
```

Delphi

```
procedure LayoutStartChanging(ASender: TObject; Operation :
LayoutChangingEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure LayoutStartChanging(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_LayoutStartChangingEvent);
begin
end;
```

Powe…

```
begin event LayoutStartChanging(long Operation)
end event LayoutStartChanging
```

VB.NET

```
Private Sub LayoutStartChanging(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_LayoutStartChangingEvent) Handles
LayoutStartChanging
End Sub
```

VB6

```
Private Sub LayoutStartChanging(ByVal Operation As
EXORGCHARTLibCtl.LayoutChangingEnum)
End Sub
```

VBA

```
Private Sub LayoutStartChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation
```

Xbas…

```
PROCEDURE OnLayoutStartChanging(oChartView,Operation)
RETURN
```

Syntax for LayoutStartChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="LayoutStartChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function LayoutStartChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComLayoutStartChanging OLELayoutChangingEnum llOperation
    Forward Send OnComLayoutStartChanging llOperation
End_Procedure
```

Visual Objects

```
METHOD OCX_LayoutStartChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_LayoutStartChanging(int _Operation)
{
}
```

XBasic

```
function LayoutStartChanging as v (Operation as
OLE::Exontrol.ChartView.1::LayoutChangingEnum)
end function
```

dBASE

```
function nativeObject_LayoutStartChanging(Operation)
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 MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the NodeFromPoint property to determine the node from the cursor. The control fires the Select event when the user clicks a node (selects a node). Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor. The AnchorClick event notifies your application that the user clicks an anchor element.

Syntax for MouseDown event, **/NET** version, on:

C#

```
private void MouseDownEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseDownEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseDownEvent
End Sub
```

Syntax for MouseDown event, **/COM** version, on:

C#

```
private void MouseDownEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_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:
AxEXORGCHARTLib._IChartViewEvents_MouseDownEvent);
begin
end;
```

Powe…

```
begin event MouseDown(integer Button,integer Shift,long X,long Y)
end event MouseDown
```

VB.NET

```
Private Sub MouseDownEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_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(oChartView,Button,Shift,X,Y)
RETURN
```

Syntax for MouseDown event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseDown(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseDown Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseDown llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseDown(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseDown(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseDown as v (Button as N,Shift as N,X as
OLE::Exontrol.ChartView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ChartView.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseDown(Button,Shift,X,Y)
return
```

The following VB sample determines the node being clicked:

```
Private Sub ChartView1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .NodeFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (n Is Nothing) Then
         MsgBox n.Caption
      End If
   End With
End Sub
```

The following C++ sample determines the node being clicked:

```
void OnMouseDownChartview1(short Button, short Shift, long X, long Y)
{
   CNode node = m_chartview.GetNodeFromPoint( X, Y );
   if ( node.m_lpDispatch != NULL )
      MessageBox( node.GetCaption() );
}
```

The following VB.NET sample determines the node being clicked:

```
Private Sub AxChartView1_MouseDownEvent(ByVal sender As Object, ByVal e As AxEXORGCHARTLib._IChartViewEvents_MouseDownEvent) Handles AxChartView1.MouseDownEvent
   With AxChartView1
      Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(e.x, e.y)
      If Not (n Is Nothing) Then
         MsgBox(n.Caption)
      End If
   End With
End Sub
```

The following C# sample determines the node being clicked:

```
private void axChartView1_MouseDownEvent(object sender, AxEXORGCHARTLib._IChartViewEvents_MouseDownEvent e)
{
```

```
    EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(e.x, e.y);
    if (node != null)
        MessageBox.Show(node.Caption);
}
```

The following VFP sample determines the node being clicked:

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

With thisform.ChartView1
    local n
    n = .NodeFromPoint(x , y )
    If !isnull(n) Then
        wait window nowait n.Caption
    EndIf
EndWith
```

event MouseMove (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user moves the mouse.

Type	Description
Button as Integer	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. Use the NodeFromPoint property to determine the node from the cursor. Use the Caption property to specify the caption of the node. Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor. The control fires the Select event when the user clicks a node. You can use the LinkCaptionFrompoint property to get the node whose caption on the link is at specified position.

Syntax for MouseMove event, **/NET** version, on:

C#

```
private void MouseMoveEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseMoveEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseMoveEvent
End Sub
```

Syntax for MouseMove event, **/COM** version, on:

C#

```
private void MouseMoveEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_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:
AxEXORGCHARTLib._IChartViewEvents_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
AxEXORGCHARTLib._IChartViewEvents_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(oChartView,Button,Shift,X,Y)
RETURN
```

Syntax for MouseMove event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseMove(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseMove(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseMove Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseMove llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseMove(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseMove as v (Button as N,Shift as N,X as
OLE::Exontrol.ChartView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ChartView.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseMove(Button,Shift,X,Y)
return
```

The following VB sample prints the caption of the node from the cursor:

```
Private Sub ChartView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
```

```
    With ChartView1
        Dim n As EXORGCHARTLibCtl.Node
        Set n = .NodeFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not (n Is Nothing) Then
            Debug.Print n.Caption
        End If
    End With
End Sub
```

The following C++ sample prints the caption of the node from the cursor:

```
void OnMouseMoveChartview1(short Button, short Shift, long X, long Y)
{
    CNode node = m_chartview.GetNodeFromPoint( X, Y );
    if ( node.m_lpDispatch != NULL )
        OutputDebugString( node.GetCaption() );
}
```

The following VB.NET sample prints the caption of the node from the cursor:

```
Private Sub AxChartView1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent) Handles
AxChartView1.MouseMoveEvent
    With AxChartView1
        Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(e.x, e.y)
        If Not (n Is Nothing) Then
            Debug.WriteLine(n.Caption)
        End If
    End With
End Sub
```

The following C# sample prints the caption of the node from the cursor:

```
private void axChartView1_MouseMoveEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_MouseMoveEvent e)
{
    EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(e.x, e.y);
    if (node != null)
        System.Diagnostics.Debug.WriteLine(node.Caption);
```

```
}
```

The following VFP sample prints the caption of the node from the cursor:

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

With thisform.ChartView1
   local n
   n = .NodeFromPoint(x , y )
   If !isnull(n) Then
      wait window nowait n.Caption
   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 DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor. The AnchorClick event notifies your application that the user clicks an anchor element.

Syntax for MouseUp event, **/NET** version, on:

C#

```
private void MouseUpEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseUpEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseUpEvent
End Sub
```

Syntax for MouseUp event, **/COM** version, on:

C#

```
private void MouseUpEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_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:
AxEXORGCHARTLib._IChartViewEvents_MouseUpEvent);
begin
end;
```

Powe…

```
begin event MouseUp(integer Button,integer Shift,long X,long Y)
end event MouseUp
```

VB.NET

```
Private Sub MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_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(oChartView,Button,Shift,X,Y)
RETURN
```

Syntax for MouseUp event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseUp(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseUp(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseUp Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseUp llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseUp(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseUp(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseUp as v (Button as N,Shift as N,X as
OLE::Exontrol.ChartView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ChartView.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseUp(Button,Shift,X,Y)
return
```

The following VB sample displays the caption of the node form the cursor:

```
Private Sub ChartView1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
```

```
Single)
   With ChartView1
      Dim n As EXORGCHARTLibCtl.Node
      Set n = .NodeFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (n Is Nothing) Then
         Debug.Print n.Caption
      End If
   End With
End Sub
```

The following C++ sample displays the caption of the node form the cursor:

```
void OnMouseUpChartview1(short Button, short Shift, long X, long Y)
{
   CNode node = m_chartview.GetNodeFromPoint( X, Y );
   if ( node.m_lpDispatch != NULL )
      OutputDebugString( node.GetCaption() );
}
```

The following VB.NET sample displays the caption of the node form the cursor:

```
Private Sub AxChartView1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_MouseUpEvent) Handles
AxChartView1.MouseUpEvent
   With AxChartView1
      Dim n As EXORGCHARTLib.Node = .get_NodeFromPoint(e.x, e.y)
      If Not (n Is Nothing) Then
         Debug.WriteLine(n.Caption)
      End If
   End With
End Sub
```

The following C# sample displays the caption of the node form the cursor:

```
private void axChartView1_MouseUpEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_MouseUpEvent e)
{
   EXORGCHARTLib.Node node = axChartView1.get_NodeFromPoint(e.x, e.y);
   if (node != null)
```

```
        System.Diagnostics.Debug.WriteLine(node.Caption);
}
```

The following VFP sample displays the caption of the node form the cursor:

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

With thisform.ChartView1
   local n
   n = .NodeFromPoint(x , y )
   If !isnull(n) Then
      wait window nowait n.Caption
   EndIf
EndWith
```

event ScrollButtonClick (ScrollBar as ScrollBarEnum, ScrollPart as ScrollPartEnum)

Occurs when the user clicks a button in the scrollbar.

Type	Description
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 ScrolPartCaption property to specify the caption of the scroll's part. Use the Background property to change the visual appearance for any part in the control's scroll bar.

Syntax for ScrollButtonClick event, **/NET** version, on:

C#

```
private void ScrollButtonClick(object
sender,exontrol.EXORGCHARTLib.ScrollBarEnum
ScrollBar,exontrol.EXORGCHARTLib.ScrollPartEnum ScrollPart)
{
}
```

VB

```
Private Sub ScrollButtonClick(ByVal sender As System.Object,ByVal ScrollBar As
exontrol.EXORGCHARTLib.ScrollBarEnum,ByVal ScrollPart As
exontrol.EXORGCHARTLib.ScrollPartEnum) Handles ScrollButtonClick
End Sub
```

Syntax for ScrollButtonClick event, **/COM** version, on:

C#

```
private void ScrollButtonClick(object sender,
AxEXORGCHARTLib._IChartViewEvents_ScrollButtonClickEvent e)
{
}
```

C++

```
void OnScrollButtonClick(long ScrollBar,long ScrollPart)
{
```

```
}
```

C++ Builder

```
void __fastcall ScrollButtonClick(TObject *Sender,Exorgchartlib_tlb::ScrollBarEnum
ScrollBar,Exorgchartlib_tlb::ScrollPartEnum ScrollPart)
{
}
```

Delphi

```
procedure ScrollButtonClick(ASender: TObject; ScrollBar :
ScrollBarEnum;ScrollPart : ScrollPartEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ScrollButtonClick(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_ScrollButtonClickEvent);
begin
end;
```

Powe…

```
begin event ScrollButtonClick(long ScrollBar,long ScrollPart)
end event ScrollButtonClick
```

VB.NET

```
Private Sub ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_ScrollButtonClickEvent) Handles
ScrollButtonClick
End Sub
```

VB6

```
Private Sub ScrollButtonClick(ByVal ScrollBar As
EXORGCHARTLibCtl.ScrollBarEnum,ByVal ScrollPart As
EXORGCHARTLibCtl.ScrollPartEnum)
End Sub
```

VBA

```
Private Sub ScrollButtonClick(ByVal ScrollBar As Long,ByVal ScrollPart As Long)
End Sub
```

VFP

```
LPARAMETERS ScrollBar,ScrollPart
```

Xbas…

```
PROCEDURE OnScrollButtonClick(oChartView,ScrollBar,ScrollPart)
RETURN
```

Syntax for ScrollButtonClick event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="ScrollButtonClick(ScrollBar,ScrollPart)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ScrollButtonClick(ScrollBar,ScrollPart)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComScrollButtonClick OLEScrollBarEnum llScrollBar
OLEScrollPartEnum llScrollPart
   Forward Send OnComScrollButtonClick llScrollBar llScrollPart
End_Procedure
```

Visual Objects
```
METHOD OCX_ScrollButtonClick(ScrollBar,ScrollPart) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ScrollButtonClick(int _ScrollBar,int _ScrollPart)
{
}
```

XBasic
```
function ScrollButtonClick as v (ScrollBar as
OLE::Exontrol.ChartView.1::ScrollBarEnum,ScrollPart as
OLE::Exontrol.ChartView.1::ScrollPartEnum)
end function
```

dBASE
```
function nativeObject_ScrollButtonClick(ScrollBar,ScrollPart)
return
```

The following VB sample displays the identifier of the scroll's button being clicked:

```
With ChartView1
    .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
    .ScrollPartCaption(exVScroll, exLeftB1Part) = "1"
    .ScrollPartCaption(exVScroll, exRightB1Part) = "2"
End With
```

```
Private Sub ChartView1_ScrollButtonClick(ByVal ScrollPart As
EXChartViewLibCtl.ScrollPartEnum)
   MsgBox (ScrollPart)
End Sub
```

The following VB.NET sample displays the identifier of the scroll's button being clicked:

```
With AxChartView1
   .set_ScrollPartVisible(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part Or
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, True)
   .set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part, "1")
   .set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, "2")
End With
```

```
Private Sub AxChartView1_ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXCHARTVIEWLib._IChartViewEvents_ScrollButtonClickEvent) Handles
AxChartView1.ScrollButtonClick
      MessageBox.Show( e.scrollPart.ToString())
End Sub
```

The following C# sample displays the identifier of the scroll's button being clicked:

```
axChartView1.set_ScrollPartVisible(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part |
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, true);
axChartView1.set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exLeftB1Part , "1");
axChartView1.set_ScrollPartCaption(EXCHARTVIEWLib.ScrollBarEnum.exVScroll,
EXCHARTVIEWLib.ScrollPartEnum.exRightB1Part, "2");
```

```
private void axChartView1_ScrollButtonClick(object sender,
AxEXCHARTVIEWLib._IChartViewEvents_ScrollButtonClickEvent e)
{
   MessageBox.Show(e.scrollPart.ToString());
}
```

The following C++ sample displays the identifier of the scroll's button being clicked:

```
m_chartView.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_chartView.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("1") );
m_chartView.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("2") );
```

```
void OnScrollButtonClickChartView1(long ScrollPart)
{
   CString strFormat;
   strFormat.Format( _T("%i"), ScrollPart );
   MessageBox( strFormat );
}
```

The following VFP sample displays the identifier of the scroll's button being clicked:

```
With thisform.ChartView1
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "1"
      .ScrollPartCaption(0, 32) = "2"
EndWith
```

event Select (OldNode as Node, NewNode as Node)

Occurs when the user selects a node.

Type	Description
OldNode as Node	A Node object that indicates the previous selected node.
NewNode as Node	A Node object that indicates the currently selected node.

Use the Select event to notify your application that user changes the selected node. Use the SelectNode property to determine the selected node. Use the Caption property to specify the caption of the node. Use the BackColor property to specify the node's background color. Use the ForeColor property to specify the node's background color. The control automatically scrolls the control's content to ensure that the node being clicked fits the control's client area, if the EnsureVisibleOnSelect property is True.

Syntax for Select event, **/NET** version, on:

C#

```
private void Select(object sender,exontrol.EXORGCHARTLib.Node
OldNode,exontrol.EXORGCHARTLib.Node NewNode)
{
}
```

VB

```
Private Sub Select(ByVal sender As System.Object,ByVal OldNode As
exontrol.EXORGCHARTLib.Node,ByVal NewNode As
exontrol.EXORGCHARTLib.Node) Handles Select
End Sub
```

Syntax for Select event, **/COM** version, on:

C#

```
private void Select(object sender,
AxEXORGCHARTLib._IChartViewEvents_SelectEvent e)
{
}
```

C++

```
void OnSelect(LPDISPATCH OldNode,LPDISPATCH NewNode)
{
}
```

C++ Builder

```
void __fastcall Select(TObject *Sender,Exorgchartlib_tlb::INode
*OldNode,Exorgchartlib_tlb::INode *NewNode)
```

```
{
}
```

Delphi

```
procedure Select(ASender: TObject; OldNode : INode;NewNode : INode);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Select(sender: System.Object; e:
AxEXORGCHARTLib._IChartViewEvents_SelectEvent);
begin
end;
```

Powe…

```
begin event Select(oleobject OldNode,oleobject NewNode)
end event Select
```

VB.NET

```
Private Sub Select(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_SelectEvent) Handles Select
End Sub
```

VB6

```
Private Sub Select(ByVal OldNode As EXORGCHARTLibCtl.INode,ByVal NewNode
As EXORGCHARTLibCtl.INode)
End Sub
```

VBA

```
Private Sub Select(ByVal OldNode As Object,ByVal NewNode As Object)
End Sub
```

VFP

```
LPARAMETERS OldNode,NewNode
```

Xbas…

```
PROCEDURE OnSelect(oChartView,OldNode,NewNode)
RETURN
```

Syntax for Select event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Select(OldNode,NewNode)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
```

```
Function Select(OldNode,NewNode)
End Function
</SCRIPT>
```

Visual Data...

```
Procedure OnComSelect Variant llOldNode Variant llNewNode
   Forward Send OnComSelect llOldNode llNewNode
End_Procedure
```

Visual Objects

```
METHOD OCX_Select(OldNode,NewNode) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Select(COM _OldNode,COM _NewNode)
{
}
```

XBasic

```
function Select as v (OldNode as OLE::Exontrol.ChartView.1::INode,NewNode as
OLE::Exontrol.ChartView.1::INode)
end function
```

dBASE

```
function nativeObject_Select(OldNode,NewNode)
return
```

The following VB sample changes the background and foreground color for the selected node:

```
Private Sub ChartView1_Select(ByVal OldNode As EXORGCHARTLibCtl.INode, ByVal
NewNode As EXORGCHARTLibCtl.INode)
   If Not (OldNode Is Nothing) Then
      With OldNode
         .ClearBackColor
         .ClearForeColor
      End With
   End If
   With NewNode
      .ForeColor = vbWhite
      .BackColor = vbBlue
   End With
```

```
End Sub
```

The following C++ sample changes the background and foreground color for the selected node:

```
void OnSelectChartview1(LPDISPATCH OldNode, LPDISPATCH NewNode)
{
   CNode oldNode( OldNode ); oldNode.m_bAutoRelease = FALSE;
   CNode newNode( NewNode ); newNode.m_bAutoRelease = FALSE;

   if ( oldNode.m_lpDispatch != NULL )
   {
      oldNode.ClearBackColor();
      oldNode.ClearForeColor();
   }
   newNode.SetBackColor( RGB(0,0,128) );
   newNode.SetForeColor( RGB(255,255,255) );
}
```

The following VB.NET sample changes the background and foreground color for the selected node:

```
Private Sub AxChartView1_SelectEvent(ByVal sender As System.Object, ByVal e As
AxEXORGCHARTLib._IChartViewEvents_SelectEvent) Handles AxChartView1.SelectEvent
   If Not (e.oldNode Is Nothing) Then
      With e.oldNode
         .ClearBackColor()
         .ClearForeColor()
      End With
   End If
   With e.newNode
      .ForeColor = ToUInt32(Color.White)
      .BackColor = ToUInt32(Color.Blue)
   End With
End Sub
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
```

```
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background and foreground color for the selected node:

```
private void axChartView1_SelectEvent(object sender,
AxEXORGCHARTLib._IChartViewEvents_SelectEvent e)
{
    if ( e.oldNode != null )
    {
        e.oldNode.ClearBackColor();
        e.oldNode.ClearForeColor();
    }
    e.newNode.BackColor = ToUInt32(Color.Blue);
    e.newNode.ForeColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background and foreground color for the selected node:

```
*** ActiveX Control Event ***
LPARAMETERS oldnode, newnode
```

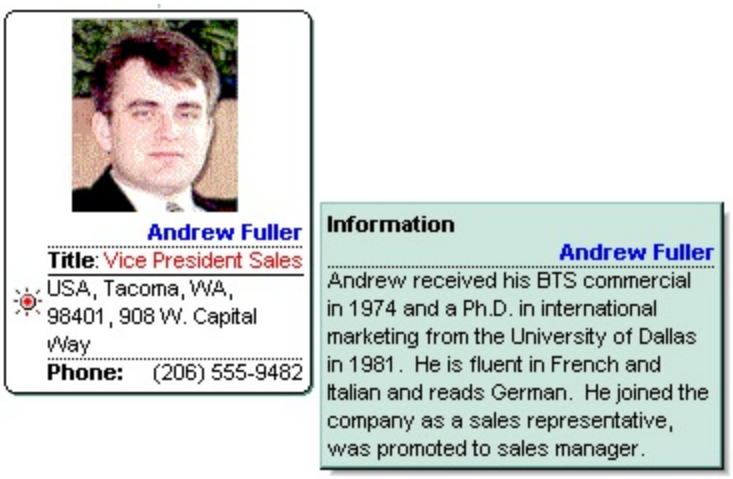
```
If !isnull(oldnode)
   With oldnode
      .ClearBackColor
      .ClearForeColor
   EndWith
EndIf
With newnode
   .ForeColor = RGB(255,255,255)
   .BackColor = RGB(0,0,128)
EndWith
```

event ToolTip (OverNode as Node, Visible as Boolean, X as Long, Y as Long, CX as Long, CY as Long)

Fired when the control prepares the object's tooltip.

Type	Description
OverNode as Node	A Node object that indicates whose tooltip is shown or hidden.
Visible as Boolean	A boolean expression that indicates whether the tooltip is shown or hidden.
X as Long	A long expression that indicates the left location of the tooltip window. The x values is always expressed in screen coordinates.
Y as Long	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 node to be shown or hidden. Use the ToolTip event to change the default position of the tooltip window. Use the ToolTip property to specify the node's tooltip. Use the ToolTip property to specify the node'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. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the NodeFromPoint property to get the node from point.



Syntax for ToolTip event, **/NET** version, on:

C#

```
private void ToolTip(object sender,exontrol.EXORGCHARTLib.Node OverNode,ref bool Visible,ref int X,ref int Y,int CX,int CY)
{
}
```

VB

```
Private Sub ToolTip(ByVal sender As System.Object,ByVal OverNode As exontrol.EXORGCHARTLib.Node,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, AxEXORGCHARTLib._IChartViewEvents_ToolTipEvent e)
{
}
```

C++

```
void OnToolTip(LPDISPATCH OverNode,BOOL FAR* Visible,long FAR* X,long FAR* Y,long CX,long CY)
{
}
```

C++ Builder

```
void __fastcall ToolTip(TObject *Sender,Exorgchartlib_tlb::INode *OverNode,VARIANT_BOOL * Visible,long * X,long * Y,long CX,long CY)
{
}
```

Delphi

```
procedure ToolTip(ASender: TObject; OverNode : INode;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: AxEXORGCHARTLib._IChartViewEvents_ToolTipEvent);
begin
end;
```

Powe…

```
begin event ToolTip(oleobject OverNode,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
AxEXORGCHARTLib._IChartViewEvents_ToolTipEvent) Handles ToolTip
End Sub
```

VB6

```
Private Sub ToolTip(ByVal OverNode As EXORGCHARTLibCtl.INode,Visible As
Boolean,X As Long,Y As Long,ByVal CX As Long,ByVal CY As Long)
End Sub
```

VBA

```
Private Sub ToolTip(ByVal OverNode As Object,Visible As Boolean,X As Long,Y As
Long,ByVal CX As Long,ByVal CY As Long)
End Sub
```

VFP

```
LPARAMETERS OverNode,Visible,X,Y,CX,CY
```

Xbas…

```
PROCEDURE OnToolTip(oChartView,OverNode,Visible,X,Y,CX,CY)
RETURN
```

Syntax for ToolTip event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ToolTip(OverNode,Visible,X,Y,CX,CY)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ToolTip(OverNode,Visible,X,Y,CX,CY)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComToolTip Variant llOverNode Boolean llVisible Integer llX Integer
llY Integer llCX Integer llCY
    Forward Send OnComToolTip llOverNode llVisible llX llY llCX llCY
End_Procedure
```

Visual Objects

```
METHOD OCX_ToolTip(OverNode,Visible,X,Y,CX,CY) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ToolTip(COM _OverNode,COMVariant /*bool*/ _Visible,COMVariant /*long*/ _X,COMVariant /*long*/ _Y,int _CX,int _CY)
{
}
```

XBasic

```
function ToolTip as v (OverNode as OLE::Exontrol.ChartView.1::INode,Visible as L,X as N,Y as N,CX as N,CY as N)
end function
```

dBASE

```
function nativeObject_ToolTip(OverNode,Visible,X,Y,CX,CY)
return
```

Expressions

An expression is a string which defines a formula or criteria, that's evaluated at runtime. The expression may be a combination of variables, constants, strings, dates and operators/functions. For instance 1000 format `` gets 1,000.00 for US format, while 1.000,00 is displayed for German format.

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

Usage examples:

- 100 + 200, adds numbers and returns 300
- "100" + 200, concatenates the strings, and returns "100200"
- currency(1000) displays the value in currency format based on the current regional setting, such as "$1,000.00" for US format.
- 1000 format `` gets 1,000.00 for English format, while 1.000,00 is displayed for German format
- 1000 format `2|.|3|,` always gets 1,000.00 no matter of settings in the control panel.
- upper("string") converts the giving string in uppercase letters, such as "STRING"
- date(dateS('3/1/' + year(9:=#1/1/2018#)) + ((1:=(((255 - 11 * (year(=:9) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(=:9) + int(year(=:9) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7)) returns the date the Easter Sunday will fall, for year 2018. In this case the expression returns #4/1/2018#. If #1/1/2018# is replaced with #1/1/2019#, the expression returns #4/21/2019#.

Listed bellow are all predefined constants, operators and functions the general-expression supports:

The constants can be represented as:

- numbers in **decimal** format (where dot character specifies the decimal separator). For instance: -1, 100, 20.45, .99 and so on
- numbers in **hexa-decimal** format (preceded by **0x** or **0X** sequence), uses sixteen distinct symbols, most often the symbols 0-9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. Hexadecimal numerals are widely used by computer system designers and programmers. As each hexadecimal digit represents four binary digits (bits), it allows a more human-friendly representation of binary-coded values. For instance, 0xFF,

0x00FF00, and so so.

- **date-time** in format **#mm/dd/yyyy hh:mm:ss#**, For instance, #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM
- **string**, if it starts / ends with any of the **'** or ` or **"** characters. If you require the starting character inside the string, it should be escaped ( preceded by a \ character ). For instance, `Mihai`, "Filimon", 'has', "\"a quote\"", and so on

The predefined constants are:

- **bias** (BIAS constant), defines the difference, in minutes, between Coordinated Universal Time (UTC) and local time. For example, Middle European Time (MET, GMT+01:00) has a time zone bias of "-60" because it is one hour ahead of UTC. Pacific Standard Time (PST, GMT-08:00) has a time zone bias of "+480" because it is eight hours behind UTC. For instance, date(value - bias/24/60) converts the UTC time to local time, or date(date('now') + bias/24/60) converts the current local time to UTC time. For instance, "date(value - bias/24/60)" converts the value date-time from UTC to local time, while "date(value + bias/24/60)" converts the local-time to UTC time.
- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

The supported binary arithmetic operators are:

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

The supported unary boolean operators are:

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

The supported binary boolean operators are:

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

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

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

The supported binary range operators, all these with the same priority 5, are :

- a **MIN** b (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- a **MAX** b (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For

instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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

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

  expression ? true_part : false_part

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

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

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

  expression array (c1,c2,c3,...cn)

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')* is equivalent with *month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'O';10:'N';11:'D')*.

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

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

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

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

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

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

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

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

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

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

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

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions. Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. The type operator may return any of the following: 0 - empty (not initialized), 1 - null, 2 - short, 3 - long, 4 - float, 5 - double, 6 - currency, **7 - date**, **8 - string**, 9 - object, 10 - error, **11 - boolean**, 12 - variant, 13 - any, 14 - decimal, 16 - char, 17 - byte, 18 - unsigned short, 19 - unsigned long, 20 - long on 64 bits, 21 - unsigned long on 64 bites. For instance *type(%1) = 8* specifies the cells (on the column with the index 1) that contains string values.
- **str** (unary operator) converts the expression to a string. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#
- **hex** (unary operator) converts the giving string from hexa-representation to a numeric value, or converts the giving numeric value to hexa-representation as string. For instance, hex(`FF`) returns 255, while the hex(255) or hex(0xFF) returns the `FF` string. The hex(hex(`FFFFFFFF`)) always returns `FFFFFFFF` string, as the second hex call converts the giving string to a number, and the first hex call converts the returned number to string representation (hexa-representation).

The bitwise operators for numbers are:

- a **bitand** b (binary operator) computes the AND operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitand 0x10111000 returns 0x00001000.
- a **bitor** b (binary operator) computes the OR operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitor 0x10111000 returns 0x11111000.
- a **bitxor** b (binary operator) computes the XOR (exclusive-OR) operation on bits of a and b, and returns the unsigned value. For instance, 0x01110010 bitxor 0x10101010 returns 0x11011000.
- a **bitshift** (b) (binary operator) shifts every bit of a value to the left if b is negative, or to the right if b is positive, for b times, and returns the unsigned value. For instance, 128 bitshift 1 returns 64 (dividing by 2) or 128 bitshift (-1) returns 256 (multiplying by

2)

- **bitnot** (unary operator) flips every bit of x, and returns the unsigned value. For instance, bitnot(0x00FF0000) returns 0xFF00FFFF.

The operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the *1000 format ''* displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

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

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of

the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.

- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

The operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- a **startwith** b (binary operator) specifies whether a string starts with specified string (

0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1

- a **endwith** b (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- a **contains** b (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- a **left** b (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- a **right** b (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- a **split** b (binary operator) splits the a using the separator b, and returns an array. For instance, the *weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' split ' '* gets the weekday as string. This operator can be used with the array.
- a **like** b (binary operator) compares the string a against the pattern b. The pattern b may contain wild-characters such as *, ?, # or [] and can have multiple patterns separated by space character. In order to have the space, or any other wild-character inside the pattern, it has to be escaped, or in other words it should be preceded by a \ character. For instance value like `F*e` matches all strings that start with F and ends on e, or value like `a* b*` indicates any strings that start with a or b character.
- a **lpad** b (binary operator) pads the value of a to the left with b padding pattern. For instance, 12 lpad "0000" generates the string "0012".
- a **rpad** b (binary operator) pads the value of a to the right with b padding pattern. For instance, 12 lpad "____" generates the string "12__".
- a **concat** b (binary operator) concatenates the a (as string) for b times. For instance, "x" concat 5, generates the string "xxxxx".

The operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"

- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The expression supports also **immediate if** (similar with iif in visual basic, or ? : in C++) ie cond ? value_true : value_false, which means that once that cond is true the value_true is used, else the value_false is used. Also, it supports variables, up to 10 from 0 to 9. For instance, 0:="Abc" means that in the variable 0 is "Abc", and =:0 means retrieves the value of the variable 0. For instance, the len(%0) ? (0:=(%1+%2) ? currency(=:0) else `` ) : `` gets the sum between second and third column in currency format if it is not zero, and only if the first column is not empty. As you can see you can use the variables to avoid computing several times the same thing (in this case the sum %1 and %2 .