ExPropertiesList

The ExPropertiesList/ExPropertiesGrid control (similar to the control used to manipulate properties in Visual Studio) provides an efficient, intuitive and visually compact way to handle data input with minimal coding and user interface design. The ExPropertiesList component is easy to use and integrate into your application. The ExPropertiesList 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.

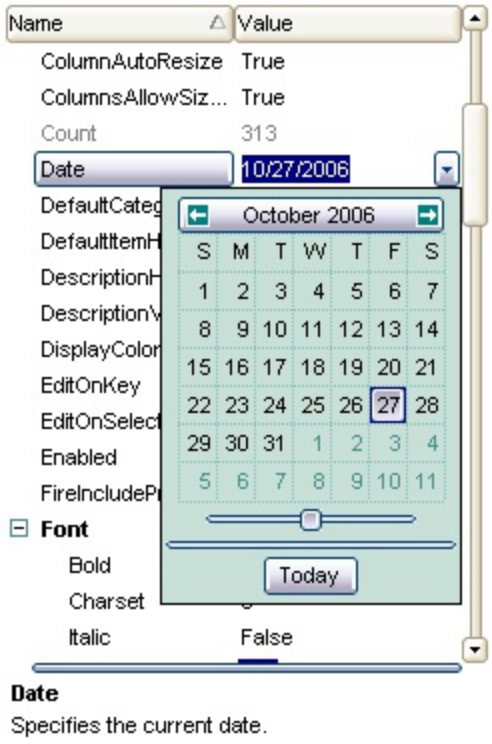
Features include:

- **Print** and Print Preview support.
- **Skinnable Interface** support (ability to apply a skin to any background part)
- Ability to browse any object that selected object contains, including **COM** or/and **.NET** objects.
- Ability to insert multiple COM or/and .NET objects in the same browser.
- Ability to add custom properties.
- Ability to include properties with **multiple** parameters, of **any** type (predefined type or not).
- Built-in **HTML** support, including icons, pictures, font and colors.
- **Filter-Prompt** support, allows you to filter the properties as you type while the filter bar is always visible on the bottom part of the list area.
- Properties of Color type (support for **EBN** skin objects)
- Tooltip Support.
- Hierarchical layout
- Browse collections and their items.
- Browse property pages.
- Browse object categories.
- Incremental search support, including expanding the object properties for looking inside.
- Ability to filter properties.
- It supports virtually all common data types including Variant, Byte, Boolean, (Long) Integer, Single, Double, Currency, Font, Icon, Picture, Date and more
- Built-in editors includes font, calendar, boolean combo, enumeration combo, colors, spin, slider, masked edit control and more

The control includes the ability to browse any COM object that exposes an implementation of IDispatch interface. For instance, any VB class provides an implementation for IDispatch interface, so the ExPropertiesList is able to browse your VB objects. Another nice feature that control provides is browsing collections and their items. If you have a collection, the ExPropertiesList can browse their items! More than that the ExPropertiesList control

expands your objects. For instance, If your object provides a property that exports another object, the ExPropertiesList control is able to browse the exported object.

All that you have to do to see it running, is to insert an instance of ExPropertiesList control to your form, and then to select the object that you want to browse, by calling Select method. That's all! If it is too much try this: PropertiesList1.Select PropertiesList1



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

How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

Specifies the alignment of the object in the source.

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

constants AppearanceEnum

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

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

constants AutoDragEnum

The AutoDragEnum type indicates what the control does when the user clicks and start dragging a row or an item. The AutoDrag property indicates the way the component supports the AutoDrag feature. The AutoDrag feature indicates what the control does when the user clicks an item and start dragging. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on.

- The flag that ends on **...OnShortTouch** indicates the action the control does when the user short touches the screen
- The flag that ends on **...OnRight** indicates the action the control does when the user right clicks the control.
- The flag that ends on **...OnLongTouch** indicates the action the control does when the user long touches the screen

The AutoDragEnum type supports the following values:

Name	Value	Description
exAutoDragNone	0	AutoDrag is disabled.
exAutoDragCopy	8	Drag and drop the selected items to a target application, and paste them as image or text. Pasting the data to the target application depends on the application. You can use the exAutoDragCopyText to specify that you want to paste as Text, or exAutoDragCopyImage as an image.
exAutoDragCopyText	9	Drag and drop the selected items to a target application, and paste them as text only. Ability to drag and drop the data as text, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop operation can start anywhere Click here to watch a movie on how exAutoDragCopyText works.
exAutoDragCopyImage	10	Drag and drop the selected items to a target application, and paste them as image only. Ability to drag and drop the data as it looks, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop

		operation can start anywhere Click here to watch a movie on how exAutoDragCopyImage works.
exAutoDragCopySnapShot	11	Drag and drop a snap shot of the current component. This option could be used to drag and drop the current snap shot of the control to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant.
exAutoDragScroll	16	The component is scrolled by clicking the item and dragging to a new position. This option can be used to allow user scroll the control's content with NO usage of the scroll bar, like on your IPhone. Ability to smoothly scroll the control's content. The feature is useful for touch screens or tables pc, so no need to click the scroll bar in order to scroll the control's content. Click here to watch a movie on how exAutoDragScroll works.
exAutoDragCopyOnShortTouch	2048	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnShortTouch	2304	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnShortTouch	2560	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnShortTouch	2816	Drag and drop a snap shot of the current component.
exAutoDragScrollOnShortTouch	4096	The component is scrolled by clicking the object and dragging to a new position.
exAutoDragCopyOnRight	524288	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnRight	589824	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnRight	655360	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnRight	720896	Drag and drop a snap shot of the current component.

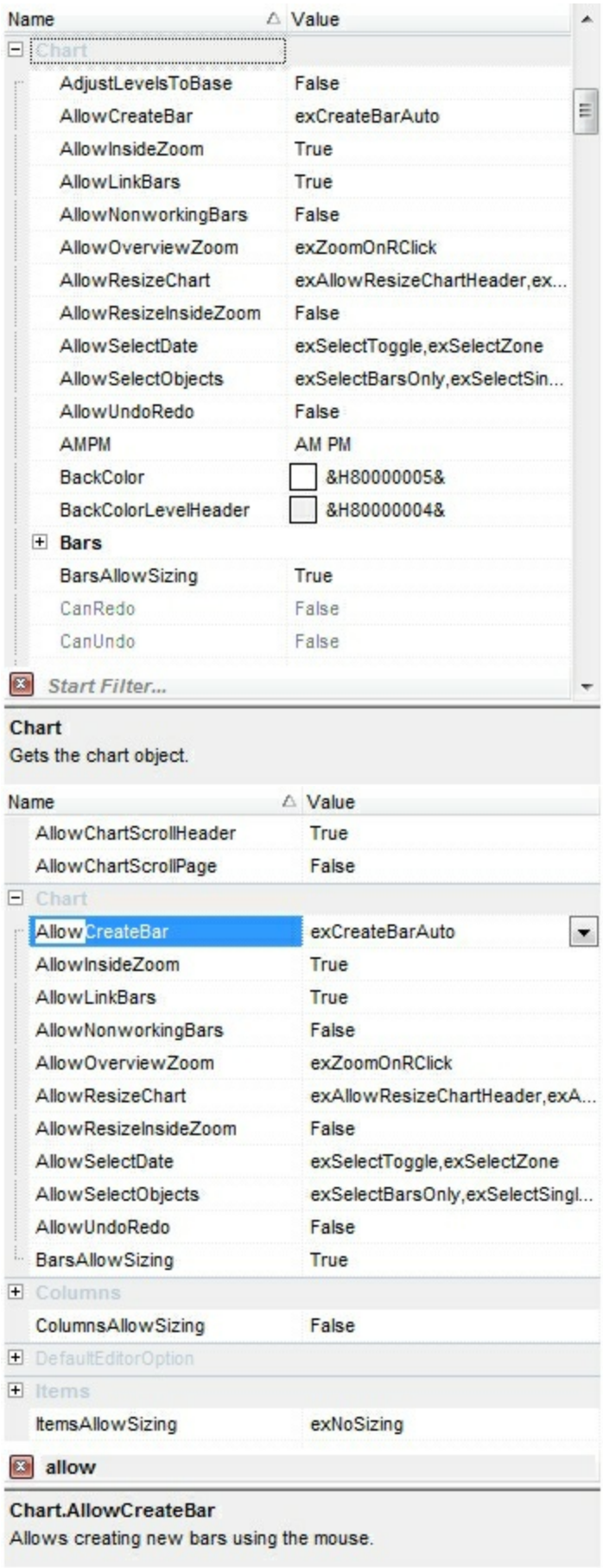
exAutoDragScrollOnRight	1048576	The component is scrolled by clicking the object and dragging to a new position.
exAutoDragCopyOnLongTouch	134217728	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnLongTouch	536994944	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnLongTouch	67772160	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnLongTouch	1845496678	Drag and drop a snap shot of the current component.
exAutoDragScrollOnLongTouch	268435456	The component is scrolled by clicking the object and dragging to a new position.

constants AutoSearchEnum

Specifies the kind of searching while user types characters within a column. Use the IncrementalSearch property to allow 'start with' incremental search or 'contains' incremental search feature in the control. For instance, if the IncrementalSearch property is exContains + exMoveOnTop, the items are re-arranged so, the first items contain the typed characters, while the rest stay unchanged. Use the ExpandOnSearch property to automatically expand parent items as user types characters. The FilterBarPromptVisible property specifies whether the control displays the control's filter prompt. The AutoSearchEnum type supports the following values.

Name	Value	Description
exStartWith	0	Defines the 'starts with' incremental search within the column. If the user type characters within the column the control looks for items that start with the typed characters. This option can be combined with the exMoveOnTop flag, which indicates that the control filters for items that start with typed characters.
exContains	1	Defines the 'contains' incremental search within the column. If the user type characters within the column the control looks for items that contain the typed characters. This option can be combined with the exMoveOnTop flag, which indicates that the control filters for items that contains typed characters.
		If this flag is present, the items being found are displayed on the top of the list. This flag can be combined with the exStartWith or exContains. The exMoveOnTop option filters for properties as you type (FilterBarPromptVisible property should be True, else it has no effect). The first screen shot shows the properties before typing anything, while the second screen shot shows the item being re-arranged on top once the user typed "Allow" (IncrementalSearch property is *exContains + exMoveOnTop*):

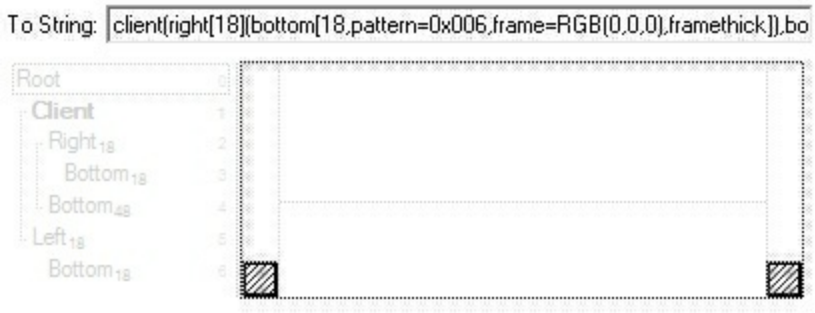
exMoveOnTop	256



In other words, if the exMoveOnTop flag is included the control filters the entries/properties that match the typed characters only.

constants BackgroundExtPropertyEnum

The BackgroundExtPropertyEnum type specifies the UI properties of the part of the EBN you can access/change at runtime. The CellBackgroundExt property specifies the EBN String format to be displayed on the cell's background. The CellBackgroundExtValue 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 CellBackgroundExt 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).*
		Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255*

exBackColorExt	1	*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 the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the*

exClientExt	2	*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. • ** ... ** 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.
exTextExt	4	• **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used. • **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold • **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with subscript

The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$

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

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

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

		instance the "<**sha**>shadow</**sha**>" generates the following picture:  or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:  *(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.*

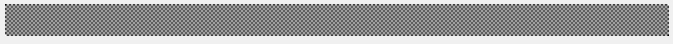
- *17, (hexa 0x11, **Middle-Center**), Text is vertically aligned in the middle, and horizontally aligned at the center.*
- *18, (hexa 0x12, **Middle-Right**), Text is vertically aligned in the middle, and horizontally aligned on the right.*
- *32, (hexa 0x20, **Bottom-Left**), Text is vertically aligned 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.*

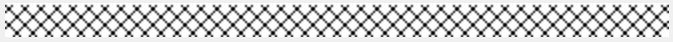
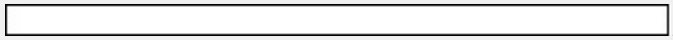
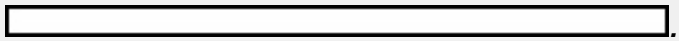
(Numeric expression)

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
exHeaderFilterBarButton	0	Specifies the background color for the drop down filter bar button.
exFooterFilterBarButton	1	Specifies the background color for the closing button in the filter bar.
exDropDownButtonUp	4	Specifies the visual appearance for the drop down button, when it is up.
exDropDownButtonDown	5	Specifies the visual appearance for the drop down button, when it is down.
exButtonUp	6	Specifies the visual appearance for the button inside the editor, when it is up.
exButtonDown	7	Specifies the visual appearance for the button inside the editor, when it is down.
exDateHeader	8	Specifies the visual appearance for the header in a

		calendar control.
exDateTodayUp	9	Specifies the visual appearance for the today button in a calendar control, when it is up.
exDateTodayDown	10	Specifies the visual appearance for the today button in a calendar control, when it is down.
exDateScrollThumb	11	Specifies the visual appearance for the scrolling thumb in a calendar control.
exDateScrollRange	12	Specifies the visual appearance for the scrolling range in a calendar control.
exDateSeparatorBar	13	Specifies the visual appearance for the separator bar in a calendar control.
exDateSelect	14	Specifies the visual appearance for the selected date in a calendar control.
exSliderRange	15	Specifies the visual appearance for the slider's bar.
exSliderThumb	16	Specifies the visual appearance for the thumb of the slider.
exSplitDesc	18	Specifies the visual appearance for the description's splitter.
exSpinUpButtonUp	22	Specifies the visual appearance for the up spin button when it is not pressed.
exSpinUpButtonDown	23	Specifies the visual appearance for the up spin button when it is pressed.
exSpinDownButtonUp	24	Specifies the visual appearance for the down spin button when it is not pressed.
exSpinDownButtonDown	25	Specifies the visual appearance for the down spin button when it is pressed.
exCursorHoverColumn	32	Specifies the visual appearance for the column when the cursor hovers the column.
exToolTipAppearance	64	Indicates the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ShowToolTip method to display a custom tooltip.

exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.
exCheckBoxState0	70	Specifies the visual appearance for the check box in 0 state (unchecked).
exCheckBoxState1	71	Specifies the visual appearance for the check box in 1 state (checked).
exCheckBoxState2	72	Specifies the visual appearance for the check box in 2 state (partial, not used).
exSpyWidget	92	Specifies the visual appearance of the widget to highlight the object from the cursor while spying.
exTreeLinesColor	186	exTreeLinesColor. Specifies the color to show the tree-lines (connecting lines from the parent to the children)
exVSUp	256	The up button in normal state.
exVSUpP	257	The up button when it is pressed.
exVSUpD	258	The up button when it is disabled.
exVSUpH	259	The up button when the cursor hovers it.
exVSThumb	260	The thumb part (exThumbPart) in normal state.
exVSThumbP	261	The thumb part (exThumbPart) when it is pressed.
exVSThumbD	262	The thumb part (exThumbPart) when it is disabled.
exVSThumbH	263	The thumb part (exThumbPart) when cursor hovers it.
exVSDown	264	The down button in normal state.
exVSDownP	265	The down button when it is pressed.
exVSDownD	266	The down button when it is disabled.
exVSDownH	267	The down button when the cursor hovers it.
exVSLower	268	The lower part (exLowerBackPart) in normal state.
exVSLowerP	269	The lower part (exLowerBackPart) when it is pressed.
exVSLowerD	270	The lower part (exLowerBackPart) when it is disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.

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

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

exScrollSizeGrip	511	Specifies the visual appearance of the control's size grip when both scrollbars are shown.

constants BorderStyleEnum

The BorderStyleEnum enumeration defines the control's border style. Use the BorderStyle property of to change the control's border.

Name	Value	Description
None	0	No border. The control has no border.
Fixed	1	Fixed. The control has a fixed size border.

constants DisplayBoolEnum

The DisplayBoolEnum type specifies the way the boolean properties displays the values. The DisplayBoolAs property specifies how the control displays the boolean properties. The DisplayBoolEnum type supports the following values:

Name	Value	Description
exBoolEnum	0	Displays the property of boolean type as enum.
exBoolCheck	1	Displays the property of boolean type as a check-box.

constants DisplayCaptionEnum

The DisplayCaptionEnum type specifies the type of captions that the DisplayCaption property returns. The DisplayCaptionEnum type supports the following values:

Name	Value	Description
exDisplayName	0	Gets the caption as displayed on Name column.
exDisplayValue	1	Gets the value as displayed on Value column.
exDisplayDescription	2	Gets the value as displayed on Description panel.
exDisplayTemplate	3	(Reserved) Gets the value in Template format.

constants DisplayColorEnum

The DisplayColorEnum type defines how the properties of color type are displayed. Use the DisplayColorEnum property to specify how the properties of color type are displayed.

Name	Value	Description
exDefault	0	Displays the property of color type as hexa like &H00FF0000&
exRGB	1	Displays the property of color type using RGB values, like RGB(255,0,0)

constants EditTypeEnum

Here's the list of supported built- in editors. Use the Add method to insert a new property to the browser. Use the Value property to assign a value to a property. Use the PropertyChange event to notify your application that the property changes its value. Use the Select method to browse for a COM object. Use the Property.Option property to specify different settings for the current editor. Use the Option property to customize the strings or behavior for different editors. Use the ModalPropertyChange event to notify your application that the user clicks the cell's button. The control supports the following type of editors:

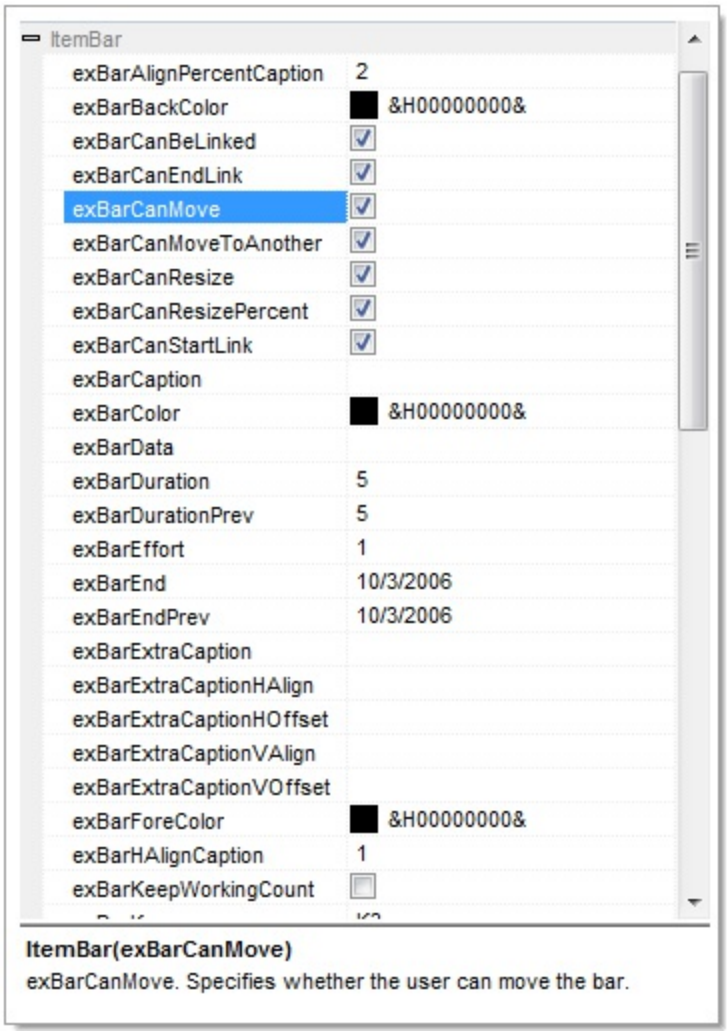
Name	Value	Description
Button	-3	Adds a button to the property, that's always visible no matter if the property is focused or selected. The EditButton or EditPage shows the button ONLY if the property is selected. The ModalPropertyChange(Property, Value, Cancel) event is fired once the user clicks the property's button (EditTypeEnum.Button type). The PropertyChange(Property) event occurs if the ModalPropertyChange event is not canceled (Cancel property is false) and the Value is changed.
Divider	-2	Adds a divider property, that merges the Name and Value columns, and displays the Name, HTMLName property on the center. The Selectable property specifies whether the user can select the property at runtime. The Sortable property specifies whether the property changes its position once the user sorts a column.
Label	-1	The property is read only (has no editor assigned) and it looks not grayed.
ReadOnly	0	The property is read only (has no editor assigned) and it looks grayed. Use the Locked property to lock a property from being changed by the user.
Edit	1	Uses a standard text box control to edit the property's value. Use the Numeric property to filter numbers of integer type. Use The NumericFloat property to filter for numbers of double type. Use the Option(exEditSingleLine) property on False to specify a multiple-lines editor.
		Provides a drop down portion that includes standard, system, or EBN colors. Use the

EditColor	2	EditColorPage type to provide your own color editor. The EBN colors are shown only if the browsed COM object exposes a VisualAppearance property like shown in the following movie.
EditFont	3	Changing the property of font types. Use the ModalPropertyChange event to notify your application that the user clicks the cell's button. The default implementation calls the common font select dialog. Use the Option property to specify different settings for the current editor.
EditFontName	4	Displays a list of fonts into your system. Provides a drop down list that includes the name of the fonts installed on your computer.
EditPicture	5	Displays the picture contained by the property. Provides a small rectangle where the picture is rendered. Use the ModalPropertyChange event to notify your application that the user clicks the cell's button. Use the Option property to specify different settings for the current editor.
EditPage	6	User can open a custom page. Use the ModalPropertyChange event to notify your application that the user clicks the cell's button.
EditBoolean	7	Displays a list with boolean values: True and False.
EditEnum	8	Presents a list of predefined values. The property accepts only one of the predefined values. The AllowMultipleValuesOnEnum property specifies whether the drop down element displays a checkbox for flags in the enumeration lists that may be a bit combination.
EditDate	9	Changing the properties of DATE type. Provides a drop down calendar control. Use the Option property to specify different settings for the current editor.
EditPassword	10	Password editor. Use the EditPassword type to mask input characters with '*' character, and to disable copy and paste inside the edit control.
EditDropDown	11	Presents a list of values. The property accepts also values that are not in the list.

EditObject	12	Specifies that the property is an object property, and the properties of the object are inserted. Use the EditObject to insert multiple COM objects to the same browser.
EditColorPage	13	Displays cells of color type and add a button to let user changes the color using custom color dialog. Use the ModalPropertyChange event to notify your application that the user clicks the cell's button.
EditCheck	14	Adds a check box entry for properties of boolean type.
EditButton	15	Adds a button and a text box to a cell. Use the ModalPropertyChange event to notify your application that the user clicks the cell's button.
EditSlider	16	Adds a slider control to the property. The SliderWidth property specify the width of the slider in the property. The SliderMin and SliderMax properties indicate the range of the values used by the slider. The SliderStep property determines the proposed change when user moves the slider. The SliderTickFrequency property specifies the frequency to display ticks on a slider control.
EditFile	17	Adds a button to select a file using the common open file dialog. Use the Option property to specify different settings for the current editor.
EditFolder	18	Adds a button to select a folder.
		Edits a property of an object. The property can have multiple parameters of any type. The ShowMultipleParams property should be on True, if the property contains multiple parameters. The property may include several child items, if one or more parameters of the property are of a predefine type such as Boolean or enumeration, that lists all possible combinations. For instance, the ItemBar property (**Items.ItemBar(Item as HITEM, Key as Variant, Property as ItemBarPropertyEnum) as Variant**) of the eXG2antt contains 3 parameters, the first two of Variant type, and the last parameter of it, of Enumeration type. The PropertiesList.Add "ItemBar", Array(G2antt1.Items, i, k), EditProperty adds a new property ItemBar to the browser and

EditProperty	19

list all available options as follows:



As you can see the exBarCanMove option is a value of ItemBarPropertyEnum as well as all child items of the ItemBar property. In other words, the EditProperty may add a single row for a property if the property has no parameters, or have no predefined parameters, or several rows, if the property returns another object, or have several parameters of a predefined type such as Boolean or Enumeration.

The following VB sample lists the ItemBar property (**Items.ItemBar(Item as HITEM, Key as Variant, Property as ItemBarPropertyEnum) as Variant**) property, for the bar from the focused item:

```
With PropertiesList1
    .Add "ItemBar", Array(G2antt1.Items,
G2antt1.Items.FocusItem,
```

```
G2antt1.Items.FirstItemBar(G2antt1.Items.FocusIten
 EditProperty
End With
```

EditPropertyWildcard	20	Reserved.
EditPropertyWildcardParent	21	Reserved.

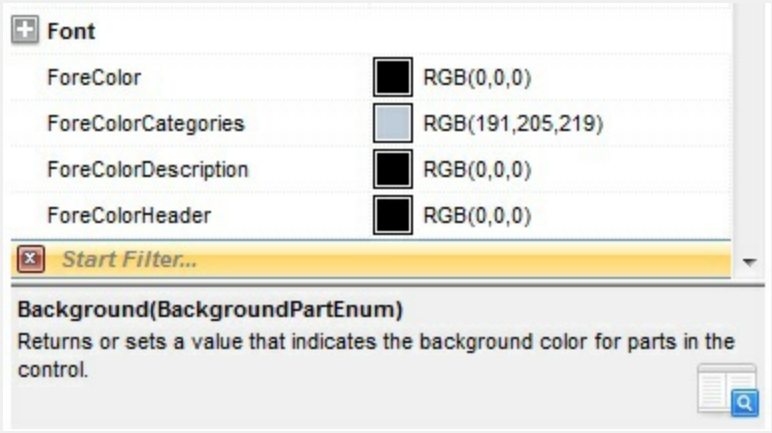
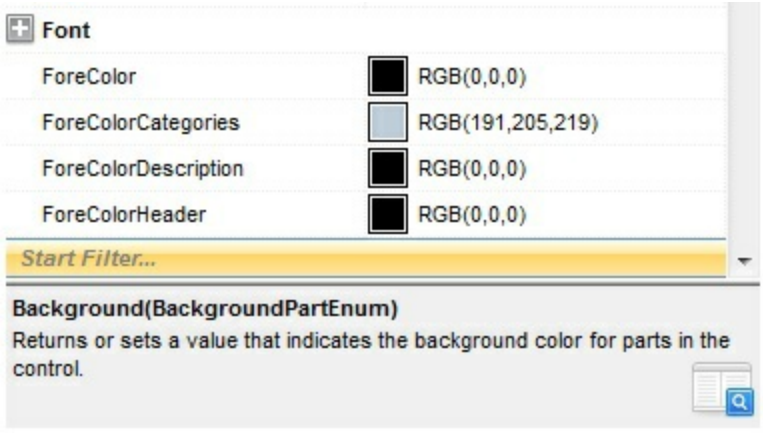
constants ExpandButtonEnum

Defines how the control displays the expanding/collapsing buttons.

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

constants FilterBarVisibleEnum

The FilterBarVisibleEnum type specifies how the control displays its filter bar prompt. The FilterBarPromptVisible property specifies whether the control's filter prompt is visible or hidden. The FilterBarVisibleEnum type specifies the following values:

Name	Value	Description
exFilterBarHidden	0	No filter prompt is shown.
exFilterBarVisible	-1	The control shows the filter bar, which displays a close button, so the user can close and remove the current filter. Use the Background(exFooterFilterBarButton) property to specify the visual appearance of the close button. 
exFilterBarAlwaysVisible	1	The control shows the filter bar with no close button. 

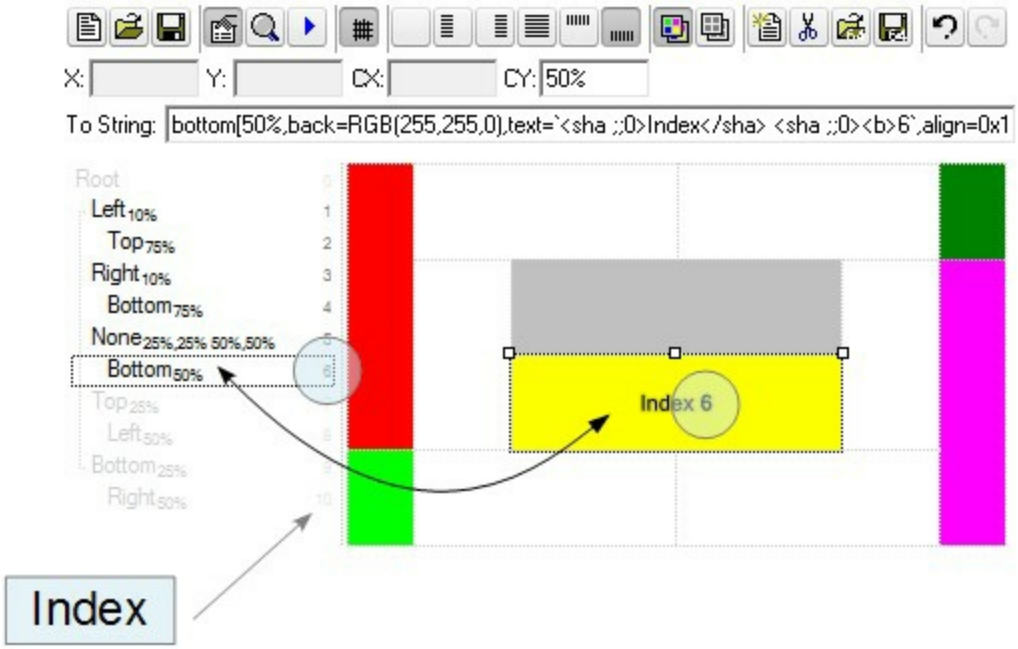
constants GridLinesEnum

Defines how the control paints the grid lines.

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

constants IndexExtEnum

The IndexExtEnum type specifies the index of the part of the EBN object to be accessed. The Index parameter of the CellBackgroundExtValue 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:*

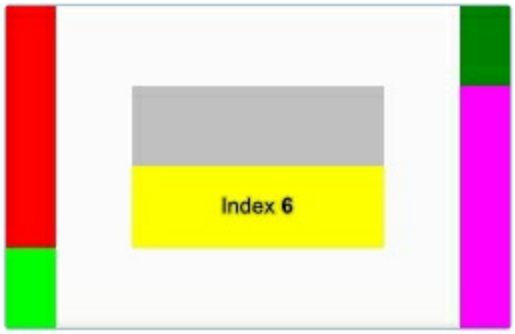


In this sample, there are 11 objects that composes the EBN, so the Index property goes from 0 which indicates the root, and 10, which is the last item in the list

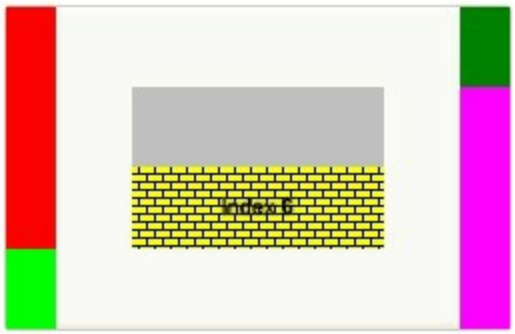
So, let's apply this format to an object, to change the exPatternExt property for the object with the Index 6:

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 OptionEnum

Specifies different options for a built-in The Option property specifies the editor's options

Name	Value	Description
exDateTodayCaption	0	Specifies the caption for the 'Today' button in the EditDate editor By default, the Option(exDateTodayCaption) is "Today". (string expression)
exDateMonths	1	Specifies the names for months to be displayed in in the EditDate editor. The list of months should be delimitated by spaces. By default, the Option(exDateMonths) = "January February March April May June July August September October November December". (string expression)
exDateWeekDays	2	Specifies the shortcut for the weekdays to be displayed in the EditDate editor. The list of shortcut for the weekdays should be separated by spaces. By default, the Option(exDateWeekDays) = "S M T W T F S". The first shortcut in the list indicates the shortcut for the Sunday, the second shortcut indicates the shortcut for Monday, and so on. (string expression)
exDateFirstWeekDay	3	Specifies the first day of the week in the EditDate editor. By default, the Option(exDateFirstWeekDay) = 0. The valid values for the Option(exDateFirstWeekDay) property are like follows: 0 - Sunday, 1 - Monday, 2 - Tuesday, 3 - Wednesday, 4 - Thursday, 5 - Friday and 6 - Saturday. (long expression, valid values are 0 to 6)
exDateShowTodayButton	4	Specifies whether the 'Today' button is visible or hidden in the EditDate editor. By default, the Option(exDateShowTodayButton) property is True. (boolean expression)
exDateMarkToday	5	Gets or sets a value that indicates whether the today date is marked in the EditDate editor. By default, Option(exDateMarkToday) property is False. (boolean expression)
exDateShowScroll	6	Specifies whether the years scroll bar is visible or hidden in the EditDate editor. By default, the Option(exDateShowScroll) property is

		True. (boolean expression)
exFontCharSet	7	Specifies a list of character sets being included in the EditFontName editor. The comma splits the characters sets in the option. By default, the exFontCharSet option is "0,2,255", that means that ANSI, OEM and SYMBOL character sets are included. (string expression) • **ANSI_CHARSET** 0 • DEFAULT_CHARSET 1 • **SYMBOL_CHARSET** 2 • MAC_CHARSET 77 • SHIFTJIS_CHARSET 128 • HANGUL_CHARSET 129 • JOHAB_CHARSET 130 • GB2312_CHARSET 134 • CHINESEBIG5_CHARSET 136 • GREEK_CHARSET 161 • TURKISH_CHARSET 162 • VIETNAMESE_CHARSET 163 • HEBREW_CHARSET 177 • ARABIC_CHARSET 178 • BALTIC_CHARSET 186 • RUSSIAN_CHARSET 204 • THAI_CHARSET 222 • EASTEUROPE_CHARSET 238 • **OEM_CHARSET** 255 For instance, if you need to include the Japanese fonts, you need to use the exFontCharSet option as "0,2,128,255", where 128 indicates the SHIFTJIS_CHARSET.
exEditFileTitle	8	Specifies the title to select a file for the EditFile editor. By default, the option is "Select File" (string expression)
exEditFileFilter	9	Specifies the filter to select files for the EditFile editor. If empty, no filter field is displayed on the open files dialog. By default, the option is "All Files (*.*)|*.*" (string expression)
exEditPictureTitle	10	Specifies the title to select a file for the EditPicture editor. By default, the option is "Load Picture"

		(string expression)
exEditPictureFilter	11	Specifies the filter to select files for the EditPicture editor. If empty, no filter field is displayed on the open files dialog. By default, the option is "All Pictures Files|*.bmp;*.dib;*.gif;*.jpg;*.wmf;*.emf;*.ico;*.cur|Bit (*.bmp;*.dib)|;*.bmp;*.dib|GIF Images (*.gif)|*.gif|JPEG Images (*.jpg)|*.jpg|Metafiles (*.wmf;*.emf)|*.wmf;*.emf|Icons (*.ico;*.cur)|*.ico;*.cur|All Files (*.*)|*.*" (string expression)
exEditFolderTitle	12	Specifies the title to select a folder for the EditFolder editor. By default, the option is "Select Folder" (string expression)
exEditFolderIncludeFiles	13	Specifies whether files are included in the EditFolder editor. By default, the option is False (Boolean expression)
exEditFolderNewUI	14	Specifies whether the EditFolder editor uses the new user interface. By default, the option is False (Boolean expression)
exEditFolderShowEditBox	15	Specifies whether the EditFolder editor displays an edit box field. By default, the option is True (Boolean expression)
exEditFolderAllowNewFolder	16	Specifies specifies whether the EditFolder editor includes a button to allow creating a new folder. By default, the option is False (Boolean expression)
exEditFolderShowPath	17	Indicates whether the EditFolder editor shows the current selected path. You can use this option in combination with exEditFolderShowEditBox on False, so the user can view the fully path of the selected file or folder in the EditFolder editor. By default, the option is False (Boolean expression)
exEditSingleLine	18	Specifies if the inside edit-box is a single or multiple lines editor. If False, you can use the exEditMaxMultipleLines option to specify the number of lines that a multiple-lines editor may display. If False, you can use the exEditAutoSizeMultipleLines option to specify if the editor is auto-sizing when user alters the editor. By default, the option is True (Boolean expression)

exEditMaxMultipleLines	19	Specifies the number of lines that a multiple-lines editor may display. This option has effect only if the exEditSingleLine property is False. By default, the option is 6 (long expression)
exEditAutoSizeMultipleLines	20	Specifies if the multiple-lines editor is auto-sizing once the user alters the property's field. This option has effect only if the exEditSingleLine property is False. • If the exEditAutoSizeMultipleLines property is **0**, the size of the editor is not changed while the user alters the editor's content. In this case, the exEditMaxMultipleLines option specifies the number of lines to be displayed while editing the field. • If the exEditAutoSizeMultipleLines property is **-1**, the size of the editor is changed while the user alters the editor's content, so it fits it content. In this case, the exEditMaxMultipleLines option specifies the number of maximum lines to be displayed while editing the field. While editing, if the number of lines grows, the size of the editor is growing too. • If the exEditAutoSizeMultipleLines property is **1**, the size of the editor is changed while the user alters the editor's content, so it fits it content. In this case, the exEditMaxMultipleLines option specifies the number of maximum lines to be displayed while editing the field. The height of the editor is not shrinking while the user is editing or removing lines. By default, the option is -1 (long expression)

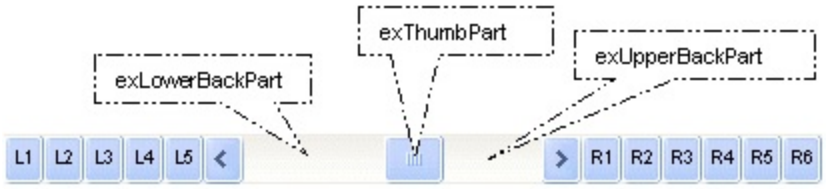
constants ScrollBarEnum

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 SortObjectsEnum

The SortObjectsEnum type specifies the position of objects to be shown on the control, when the user sorts a column. The SortObjects property specifies whether the objects are placed on top or bottom side of the control when the user sorts a column. The SortObjectsEnum type supports the following values.

Name	Value	Description
exSortObjectsDefault	0	Default sorting. The object properties are placed on their sorting position.
exSortObjectsTop	1	The object properties are put on the top of the list.
exSortObjectsBottom	2	The object properties are put on the bottom of the list.

constants SortOnClickEnum

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

Name	Value	Description
exNoSort	0	The column is not sorted when the user clicks the column's header
exDefaultSort	-1	The control sorts the column when the user clicks the column's header

constants ToStringEnum

The ToString method gets the list of properties with their values as they are displayed in the control.

Name	Value	Description
exLiterals	0	Generates the literals, as in the type library.
exNumbers	1	Generates the numbers instead the literals.

constants UIVisualThemeEnum

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

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

Appearance object

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

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

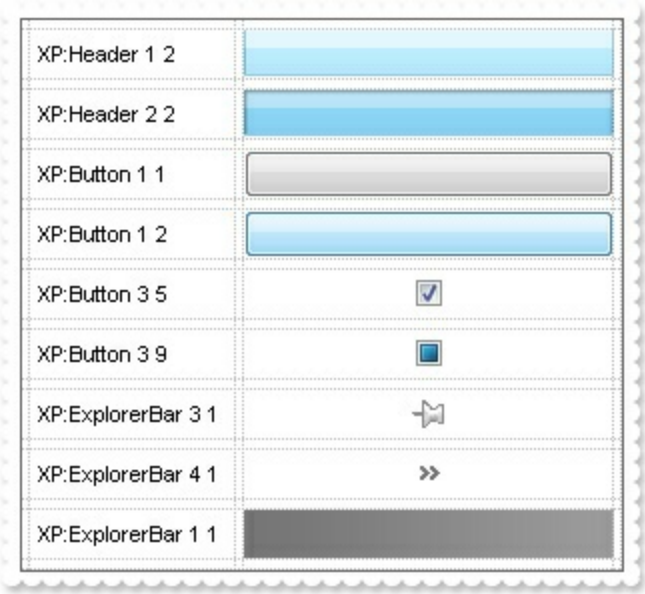
method Appearance.Add (ID as Long, Skin as Variant)

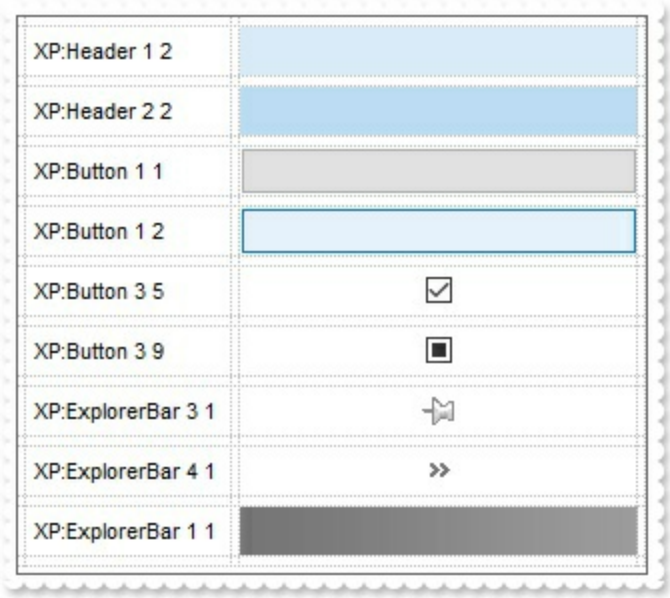
Adds or replaces a skin object to the control.

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

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

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



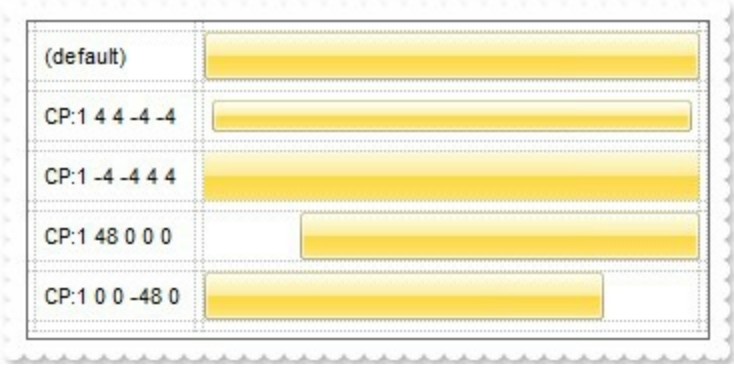


Skin as Variant

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

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

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

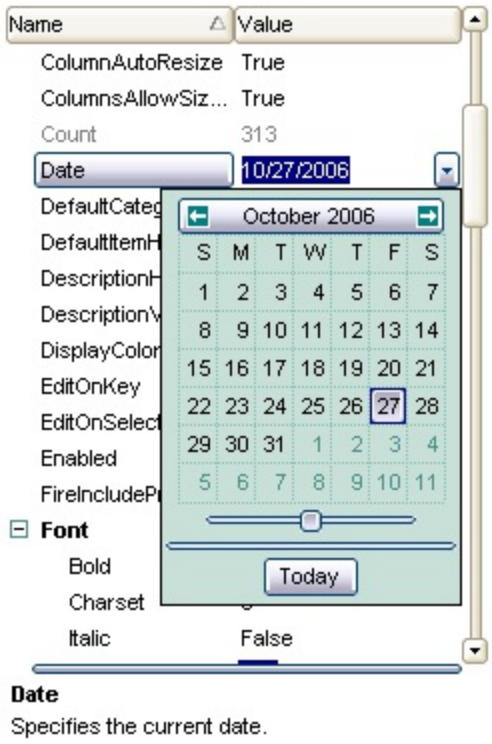


Return	Description
Boolean	A Boolean expression that indicates whether the new skin was added or replaced.

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control, when the "XP:" prefix is not specified in the Skin parameter (available for Windows XP systems). 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 do multiple changes to the control. Use the Refresh method to refresh the control. Use the Background property to specify the visual appearance for the parts in the control.

The identifier you choose for the skin is very important to be used in the

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



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

Control/ClassName	Part	States
BUTTON	BP_CHECKBOX = 3	CBS_UNCHECKED 1 CBS_UNCHECKE CBS_UNCHECKED = 3 CBS_UNCHECKED = 4 CBS_CHECKE 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_CHECKEDDI CBS_MIXEDNORM CBS_MIXEDHOT =

		CBS_MIXEDPRESS CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
	BP_PUSHBUTTON = 1	PBS_NORMAL = 1 = 2 PBS_PRESSED PBS_DISABLED = PBS_DEFAULTED =
	BP_RADIOBUTTON = 2	RBS_UNCHECKED 1 RBS_UNCHECKE RBS_UNCHECKED = 3 RBS_UNCHECKED = 4 RBS_CHECKED 5 RBS_CHECKEDH RBS_CHECKEDPR RBS_CHECKEDDIS
	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1
COMBOBOX	CP_DROPDOWNBUTTON = 1	CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED =
EDIT	EP_CARET = 2	
	EP_EDITTEXT = 1	ETS_NORMAL = 1 2 ETS_SELECTED ETS_DISABLED = ETS_FOCUSED = 5 ETS_READONLY = ETS_ASSIST = 7
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
	EBP_HEADERCLOSE = 2	EBHC_NORMAL = EBHC_HOT = 2 EBHC_PRESSED =
	EBP_HEADERPIN = 3	EBHP_NORMAL = EBHP_HOT = 2 EBHP_PRESSED = EBHP_SELECTEDN 4 EBHP_SELECTE EBHP_SELECTEDP 6

	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSE
	EBP_NORMALGROUPBACKGROUND = 5	
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_NORMAL EBNGC_HOT = 2 EBNGC_PRESSED
	EBP_NORMALGROUPEXPAND = 7	EBNGE_NORMAL EBNGE_HOT = 2 EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
	EBP_SPECIALGROUPCOLLAPSE = 10	EBSGC_NORMAL EBSGC_HOT = 2 EBSGC_PRESSED
	EBP_SPECIALGROUPEXPAND = 11	EBSGE_NORMAL EBSGE_HOT = 2 EBSGE_PRESSED
	EBP_SPECIALGROUPHEAD = 12	
HEADER	HP_HEADERITEM = 1	HIS_NORMAL = 1 H 2 HIS_PRESSED =
	HP_HEADERITEMLEFT = 2	HILS_NORMAL = 1 = 2 HILS_PRESSE
	HP_HEADERITEMRIGHT = 3	HIRS_NORMAL = 1 = 2 HIRS_PRESSE
	HP_HEADERSORTARROW = 4	HSAS_SORTEDUP HSAS_SORTEDDO
LISTVIEW	LVP_EMPTYTEXT = 5	
	LVP_LISTDETAIL = 3	
	LVP_LISTGROUP = 2	
	LVP_LISTITEM = 1	LIS_NORMAL = 1 L 2 LIS_SELECTED = LIS_DISABLED = 4 LIS_SELECTEDNO 5
	LVP_LISTSORTEDDETAIL = 4	
MENU	MP_MENUBARDROPDOWN = 4	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
		MS_NORMAL = 1 MS_SELECTED = 2

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

	RP_CHEVRON = 4	CHEVS_HOT = 2 CHEVS_PRESSED
	RP_CHEVRONVERT = 5	
	RP_GRIPPER = 1	
	RP_GRIPPERVERT = 2	
SCROLLBAR	SBP_ARROWBTN = 1	ABS_DOWNDISAB ABS_DOWNHOT, ABS_DOWNNORM ABS_DOWNPRESS ABS_UPDISABLED ABS_UPHOT, ABS_UPNORMAL, ABS_UPPRESSED, ABS_LEFTDISABLI ABS_LEFTHOT, ABS_LEFTNORMA ABS_LEFTPRESSE ABS_RIGHTDISAB ABS_RIGHTHOT, ABS_RIGHTNORM ABS_RIGHTPRESS
	SBP_GRIPPERHORZ = 8	
	SBP_GRIPPERVERT = 9	
	SBP_LOWERTRACKHORZ = 4	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_LOWERTRACKVERT = 6	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNHORZ = 2	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNVERT = 3	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
		SCRBS_NORMAL = SCRBS_HOT = 2

	SBP_UPPERTRACKHORZ = 5	SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKVERT = 7	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_SIZEBOX = 10	SZB_RIGHTALIGN SZB_LEFTALIGN =
SPIN	SPNP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	SPNP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	SPNP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	SPNP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
STARTPANEL	SPP_LOGOFF = 8	
	SPP_LOGOFFBUTTONS = 9	SPLS_NORMAL = SPLS_HOT = 2 SPLS_PRESSED =
	SPP_MOREPROGRAMS = 2	
	SPP_MOREPROGRAMSARROW = 3	SPS_NORMAL = 1 = 2 SPS_PRESSED
	SPP_PLACESLIST = 6	
	SPP_PLACESLISTSEPARATOR = 7	
	SPP_PREVIEW = 11	
	SPP_PROGLIST = 4	
	SPP_PROGLISTSEPARATOR = 5	
	SPP_USERPANE = 1	
	SPP_USERPICTURE = 10	
STATUS	SP_GRIPPER = 3	
	SP_PANE = 1	
	SP_GRIPPERPANE = 2	

TAB	TABP_BODY = 10	
	TABP_PANE = 9	
	TABP_TABITEM = 1	TIS_NORMAL = 1 T 2 TIS_SELECTED = TIS_DISABLED = 4 TIS_FOCUSED = 5
	TABP_TABITEMBOTHEDGE = 4	TIBES_NORMAL = TIBES_HOT = 2 TIBES_SELECTED TIBES_DISABLED TIBES_FOCUSED =
	TABP_TABITEMLEFTEDGE = 2	TILES_NORMAL = TILES_HOT = 2 TILES_SELECTED TILES_DISABLED = TILES_FOCUSED =
	TABP_TABITEMRIGHTEDGE = 3	TIRES_NORMAL = TIRES_HOT = 2 TIRES_SELECTED TIRES_DISABLED TIRES_FOCUSED
	TABP_TOPTABITEM = 5	TTIS_NORMAL = 1 = 2 TTIS_SELECTE TTIS_DISABLED = TTIS_FOCUSED =
	TABP_TOPTABITEMBOTHEDGE = 8	TTIBES_NORMAL = TTIBES_HOT = 2 TTIBES_SELECTE TTIBES_DISABLED TTIBES_FOCUSED
	TABP_TOPTABITEMLEFTEDGE = 6	TTILES_NORMAL = TTILES_HOT = 2 TTILES_SELECTE TTILES_DISABLED TTILES_FOCUSED
	TABP_TOPTABITEMRIGHTEDGE = 7	TTIRES_NORMAL TTIRES_HOT = 2 TTIRES_SELECTE TTIRES_DISABLED TTIRES_FOCUSED
TASKBAND	TDP_GROUPCOUNT = 1	

	TDP_FLASHBUTTON = 2	
	TDP_FLASHBUTTONGROUPMENU = 3	
TASKBAR	TBP_BACKGROUNDBOTTOM = 1	
	TBP_BACKGROUNDLEFT = 4	
	TBP_BACKGROUNDRIGHT = 2	
	TBP_BACKGROUNDTOP = 3	
	TBP_SIZINGBARBOTTOM = 5	
	TBP_SIZINGBARBOTTOMLEFT = 8	
	TBP_SIZINGBARRIGHT = 6	
	TBP_SIZINGBARTOP = 7	
TOOLBAR	TP_BUTTON = 1	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_DROPDOWNBUTTON = 2	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTON = 3	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTONDROPDOWN = 4	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATOR = 5	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATORVERT = 6	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED

TOOLTIP	TTP_BALLOON = 3	TTBS_NORMAL = 1 TTBS_LINK = 2
	TTP_BALLOONTITLE = 4	TTBS_NORMAL = 1 TTBS_LINK = 2
	TTP_CLOSE = 5	TTCS_NORMAL = 1 TTCS_HOT = 2 TTCS_PRESSED = 3
	TTP_STANDARD = 1	TTSS_NORMAL = 1 TTSS_LINK = 2
	TTP_STANDARDTITLE = 2	TTSS_NORMAL = 1 TTSS_LINK = 2
TRACKBAR	TKP_THUMB = 3	TUS_NORMAL = 1 2 TUS_PRESSED = 3 TUS_FOCUSED = 4 TUS_DISABLED = 5
	TKP_THUMBBOTTOM = 4	TUBS_NORMAL = 1 TUBS_HOT = 2 TUBS_PRESSED = 3 TUBS_FOCUSED = 4 TUBS_DISABLED = 5
	TKP_THUMBLEFT = 7	TUVLS_NORMAL = 1 TUVLS_HOT = 2 TUVLS_PRESSED 3 TUVLS_FOCUSED 4 TUVLS_DISABLED 5
	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = 1 TUVRS_HOT = 2 TUVRS_PRESSED 3 TUVRS_FOCUSED 4 TUVRS_DISABLED 5
	TKP_THUMBTOP = 5	TUTS_NORMAL = 1 TUTS_HOT = 2 TUTS_PRESSED = 3 TUTS_FOCUSED = 4 TUTS_DISABLED = 5
	TKP_THUMBVERT = 6	TUVS_NORMAL = 1 TUVS_HOT = 2 TUVS_PRESSED = 3 TUVS_FOCUSED = 4 TUVS_DISABLED = 5
	TKP_TICS = 9	TSS_NORMAL = 1

	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	TNP_ANIMBACKGROUND = 2	
	TNP_BACKGROUND = 1	
TREEVIEW	TVP_BRANCH = 3	
	TVP_GLYPH = 2	GLPS_CLOSED = 1 GLPS_OPENED = 2
	TVP_TREEITEM = 1	TREIS_NORMAL = 1 TREIS_HOT = 2 TREIS_SELECTED = 3 TREIS_DISABLED = 4 TREIS_SELECTEDNOTFOCUS = 5
WINDOW	WP_CAPTION = 1	CS_ACTIVE = 1 CS_INACTIVE = 2 CS_DISABLED = 3
	WP_CAPTIONSIZINGTEMPLATE = 30	
	WP_CLOSEBUTTON = 18	CBS_NORMAL = 1 CBS_HOT = 2 CBS_PUSHED = 3 CBS_DISABLED = 4
	WP_DIALOG = 29	
	WP_FRAMEBOTTOM = 9	FS_ACTIVE = 1 FS_INACTIVE = 2
	WP_FRAMEBOTTOMSIZINGTEMPLATE = 36	
	WP_FRAMELEFT = 7	FS_ACTIVE = 1 FS_INACTIVE = 2
	WP_FRAMELEFTSIZINGTEMPLATE = 32	
	WP_FRAMERIGHT = 8	FS_ACTIVE = 1 FS_INACTIVE = 2
	WP_FRAMERIGHTSIZINGTEMPLATE = 34	
	WP_HELPBUTTON = 23	HBS_NORMAL = 1 HBS_HOT = 2 HBS_PUSHED = 3 HBS_DISABLED = 4
	WP_HORZSCROLL = 25	HSS_NORMAL = 1 HSS_HOT = 2 HSS_PUSHED = 3 HSS_DISABLED = 4
	WP_HORZTHUMB = 26	HTS_NORMAL = 1 HTS_HOT = 2 HTS_PUSHED = 3 HTS_DISABLED = 4

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

	= 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37	
WP_SMALLFRAMELEFT = 10	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMELEFTSIZINGTEMPLATE = 33	
WP_SMALLFRAMERIGHT = 11	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35	
WP_SMALLHELPBUTTON	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_VERTSCROLL = 27	VSS_NORMAL = 1 = 2 VSS_PUSHED VSS_DISABLED =
WP_VERTTHUMB = 28	VTS_NORMAL = 1 2 VTS_PUSHED = VTS_DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

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

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

property Appearance.RenderType as Long

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

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

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

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

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

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

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

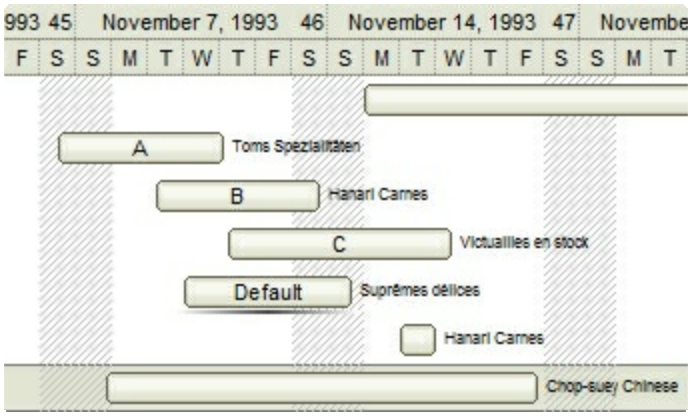
In the following screen shot the following objects displays the current EBN with a different color:

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

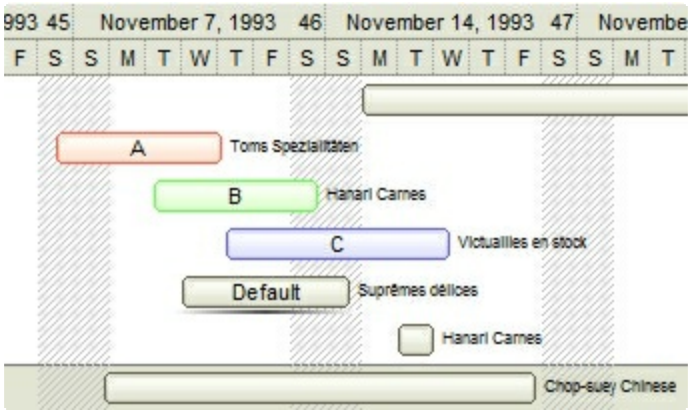
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

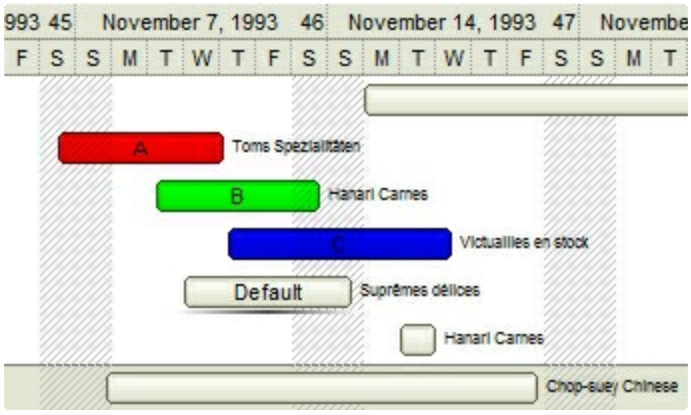
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



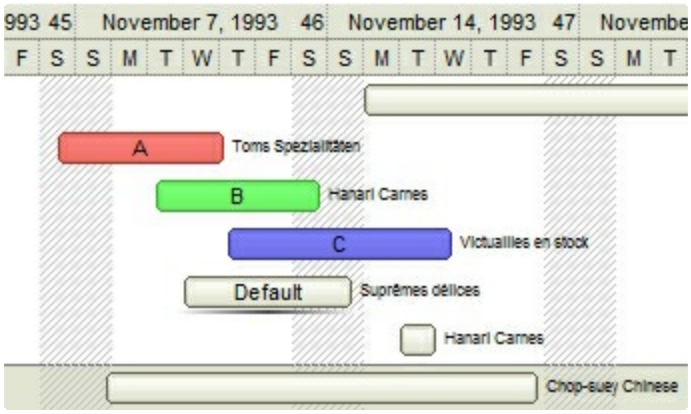
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the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



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

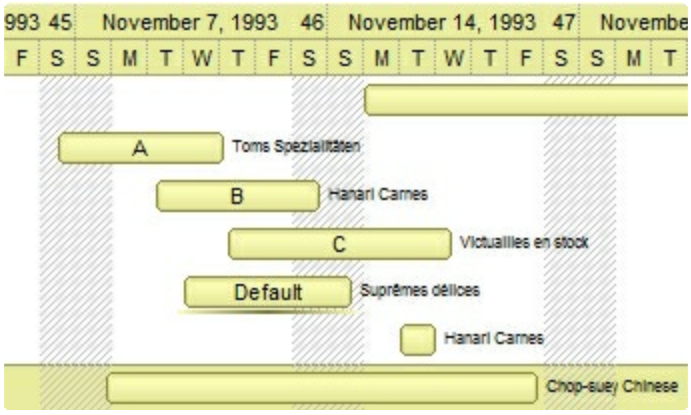


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

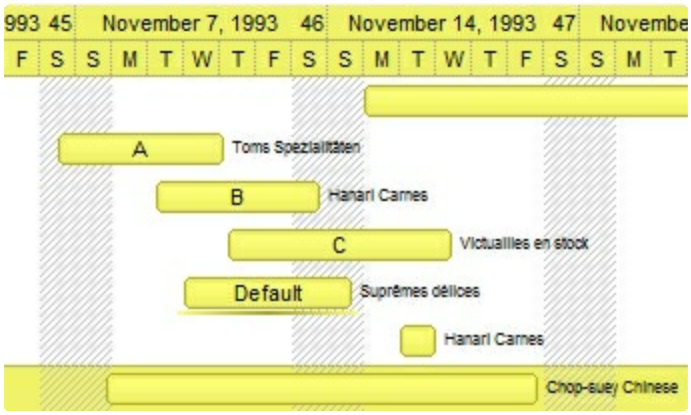


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

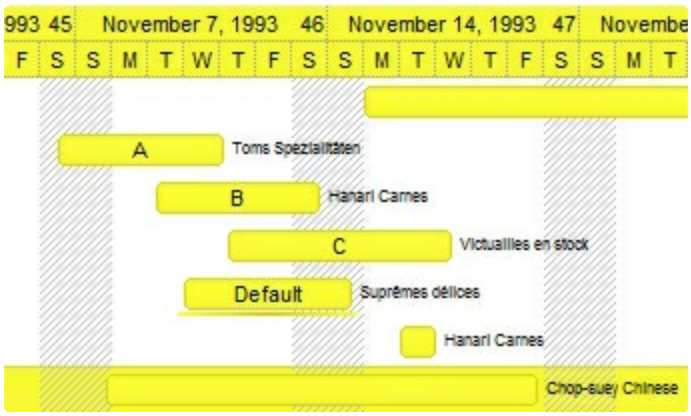
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



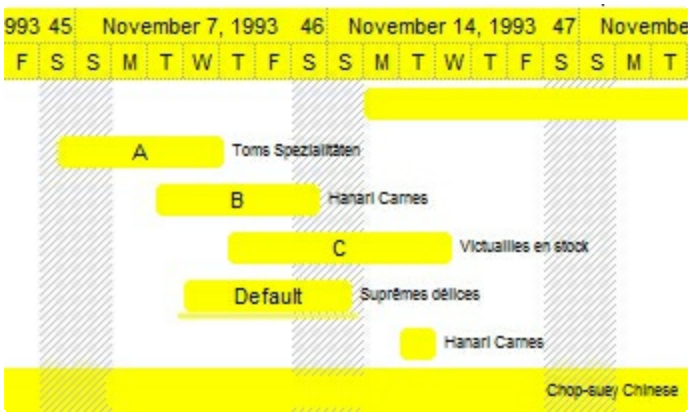
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



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



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


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

PropertiesList 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: {A703DF80-DFF3-48D7-A4C7-47CF6A48425C}. The object's program identifier is: "Exontrol.PropertiesList". The /COM object module is: "ExPropertiesList.dll"

The ExPropertiesList control (similar to the control used to manipulate properties in Visual Studio) provides an efficient, intuitive and visually compact way to handle data input with minimal coding and user interface design. A built from the ground up using 100% ATL-based code, the ExPropertiesList represents some of the most advanced properties list technology available in the ActiveX marketplace. The ExPropertiesList ActiveX control is easy to use and too easy to integrate into your application. The doesn't require any additional libraries, like MFC library.

The control includes the ability to browse any COM object that exposes an implementation of IDispatch interface. For instance, any VB class provides an implementation for IDispatch interface, so the ExPropertiesList is able to browse your VB objects. Another nice feature that control provides is browsing collections and their items. If you have a collection, the ExPropertiesList can browse their items! More than that the ExPropertiesList control expands your objects. For instance, If your object provides a property that exports another object, the ExPropertiesList control is able to browse the exported object Here's the list of supported properties and methods:

Name	Description
Add	Adds a custom entry to the list.
AllowDrop	Gets or sets a value indicating whether the control can accept data that the user drags into it.
AllowDuplicateEntries	Specifies whether the Add method allows adding new properties with the same caption on the Name column.
AllowMultipleValuesOnEnum	Specifies whether the enum types display bit combination of predefined values.
AllowSpin	Retrieves or sets a value that indicates whether the component uses a spin control to edit numeric values.
AllowSpy	Specifies whether the control can spy other UI components or parts of them.
AllowSpyOn	Specifies handle of the window where the spy can find UI objects.
AllowTooltip	Specifies whether the control displays a tooltip when the string value is too long.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
	Attaches a script to the current object, including the

AttachTemplate	events, from a string, file, a safe array of bytes.
AutoDrag	Gets or sets a value that indicates the way the component supports the AutoDrag feature.
AutoIndent	Specifies a value that indicates whether child items are automatically indented.
BackColor	Retrieves or sets a value that indicates the control's background color.
BackColorAlternate	Specifies the background color used to display alternate items in the control.
BackColorCategories	Specifies the category items background color.
BackColorDescription	Specifies the description's background color.
BackColorHeader	Specifies the header's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BorderStyle	Retrieves or sets the border style of the control.
CaptionMessageBox	Specifies the caption to be displayed on the message box, in case the user inputs an invalid value.
Clear	Clears the control's content.
ColumnAutoResize	Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.
ColumnCaption	Retrieves or sets the column's caption.
ColumnsAllowSizing	Retrieves or sets a value that indicates whether a user can resize columns at run-time.
ColumnWidth	Retrieves or sets the column's width.
Copy	Copies the control's content to the clipboard, in the EMF format.
CopyTo	Exports the control's view to an EMF, PDF, BMP, PNG, GIF, TIF file.
Count	Counts the properties in the control.
DefaultCategory	Retrieves or sets the default category.
DefaultItemHeight	Retrieves or sets the default item height.

DescriptionHeight	Retrieves or sets a value that indicates the height of the description area.
DescriptionVisible	Retrieves or sets a value that indicates whether the description is visible or hidden.
DisplayBoolAs	Specifies how the properties of boolean type are displayed.
DisplayColorAs	Specifies how the properties of color type are displayed.
EditOnKey	Customizes the F4 key to let user edits a property using the keys.
EditOnSelect	Retrieves or sets a value that indicates whether the properties browser is ready to edit a value when the selection is changed.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
ExpandAll	Expands all items.
ExpandItem	Expands or collapses an item.
ExpandOnSearch	Expands items automatically while user types characters to search for a specific property.
FilterBarFont	Retrieves or sets the font for control's filter bar.
FilterBarPrompt	Specifies the caption to be displayed when the filter pattern is missing.
FilterBarPromptPattern	Specifies the pattern for the filter prompt.
FilterBarPromptVisible	Shows or hides the filter prompt.
FireIncludeProperty	Retrieves or sets a value that indicates whether the IncludeProperty event is fired.
Font	Retrieves or sets the control's font.
ForeColor	Retrieves or sets a value that indicates the control's foreground color.
ForeColorCategories	Specifies the category items foreground color.
ForeColorDescription	Specifies the description's foreground color.

ForeColorHeader	Specifies the header's foreground color.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
GridLineColor	Retrieves or sets the grid line color.
HasButtons	Adds a button to the left side of each parent item. The user can click the button to expand or collapse the child items as an alternative to double-clicking the parent item.
HasButtonsCustom	Specifies the index of icons for +/- signs when the HasButtons property is exCustom.
HasGridLines	Retrieves or sets a value that indicates whether the grid lines are visible or hidden.
HasLines	Enhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their corresponding parent item.
HeaderAppearance	Retrieves or sets a value that indicates the header's appearance.
HeaderHeight	Retrieves or sets a value indicating the control's header height.
HeaderVisible	Retrieves or sets a value that indicates whether the control's header is visible or hidden.
HideSelection	Specifies whether selected property appears selected when the control loses focus.
HotBackColor	Retrieves or sets a value that indicates the hot-tracking background color.
HotForeColor	Retrieves or sets a value that indicates the hot-tracking foreground color.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets the control's image list at runtime.
ImageSize	Retrieves or sets the size of icons the control displays.
IncrementalSearch	Specifies whether the incremental search feature looks for starting of the property or if it contains the typed characters.
Indent	Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
	Retrieves or sets a value that indicates the base index

IndexItemsCollection	when control enumerates the items in the collection.
Interfaces	Retrieves the interfaces implemented by the object.
InvalidValueMessage	Retrieves or sets a value that indicates the error message displayed by browser when changing property's value fails. No error message occurs if is empty.
Item	Returns a Property object based on its index.
Layout	Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.
LinkCategories	Retrieves or sets a value that indicates whether the categories are linked.
MarkCategories	Specifies whether the object's categories are splited by separator lines
MarkLineColor	Retrieves or sets a value that indicates the color of line that splits the categories.
NameItemsCollection	Retrieves or sets a list of property's names separated by semicolon (;), that are used by properties browser when it requires a name for an item into a collection.
Option	Specifies an option for the editor.
Property	Gets a Property object given property's name or property's identifier.
ReadOnly	Gets or sets whether the properties browser is read-only.
Refresh	Refreshes the properties values.
Remove	Removes a property from the list.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
ScrollButtonHeight	Specifies the height of the button in the vertical scrollbar.
ScrollButtonWidth	Specifies the width of the button in the horizontal scrollbar.
ScrollFont	Retrieves or sets the scrollbar's font.
ScrollHeight	Specifies the height of the horizontal scrollbar.
ScrollOrderParts	Specifies the order of the buttons in the scroll bar.
ScrollPartCaption	Specifies the caption being displayed on the specified scroll part.
ScrollPartCaptionAlignment	Specifies the alignment of the caption in the part of the scroll bar.

ScrollPartEnable	Indicates whether the specified scroll part is enabled or disabled.
ScrollPartVisible	Indicates whether the specified scroll part is visible or hidden.
ScrollThumbSize	Specifies the size of the thumb in the scrollbar.
ScrollToolTip	Specifies the tooltip being shown when the user moves the scroll box.
ScrollWidth	Specifies the width of the vertical scrollbar.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
Select	Browses a new object to control.
SelectedObject	Browses a new object (com or .net) in the control.
SelectedProperty	Specifies the selected property.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowCategories	Retrieves or sets a value whether the browser includes the object categories.
ShowHidden	Retrieves or sets a value that indicates whether the properties browser displays the hidden members.
ShowItemsCollection	Retrieves or sets a value that indicates whether the properties browser includes the elements of a property that contains a collection.
ShowMultipleParams	Specifies whether the control loads properties with multiple parameters.
ShowNonBrowsable	Retrieves or sets a value that indicates whether the control displays the non browseable members.
ShowObjects	Retrieves or sets a value that indicates whether the properties browser includes the properties of object type.
ShowPropertyPages	Retrieves or sets a value that indicates whether the properties browser displays the object property pages.
ShowReadOnly	Retrieves or sets a value that indicates whether the properties browser displays the read only properties.
ShowRestricted	Retrieves or sets a value that indicates whether the properties browse displays the restricted members.
ShowToolTip	Shows the specified tooltip at given position.

ShowVariables	Retrieves or sets a value that indicates whether the control displays the object variables. An object of IFontDisp type has variables like: Name, Size, ...
Sort	Sorts the control.
SortObjects	Specifies how the object properties are positioned once a Sort occurs.
SortOnClick	Retrieves or sets a value that indicates whether the control sorts automatically the data when the user click on column's caption.
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.
ToString	Saves the control's content to a string, as it is displayed.
UseVisualTheme	Specifies whether the control uses the current visual theme to display certain UI parts.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.
VisualDesign	Invokes the control's VisualAppearance designer.

method PropertiesList.Add (Property as String, Value as Variant, Type as EditTypeEnum, [Description as Variant], [Parent as Variant], [Template as Variant])

Adds a custom property to the list.

Type	Description
Property as String	A string expression that indicates the property's name.
Value as Variant	A Variant value that indicates the property's value.
Type as EditTypeEnum	An EditTypeEnum expression that indicates the property's built-in editor.
Description as Variant	A string expression that indicates the property's description.
Parent as Variant	A string expression that indicates the name of the parent property, or a long expression that indicates the identifier of the parent property.
Template as Variant	Reserved.

Return	Description
Property	A Property object that represents the newly created object. Use Property property to get a property based on its name or based on its identifier.

The Add method adds a new property to the current list. The Add method associates to the newly created Property object an identifier, that's unique. Use the ID property to get the property's identifier. Use the Remove property to remove a property from the collection. Use the UserData property to associate an extra data to a property. Use the PropertyChange event to notify your application when a property's value is changed. Use the AddValue method to add new values to a drop down editor (EditEnum, EditDropDown types). Use the Select method to browse for a new COM object. Use the HTMLName property to assign icons, pictures, font attributes or colors, to parts of the caption being displayed in the Name column. Use the Option property to specify custom settings for editor assigned to a property. If the AllowDuplicateEntries property is False, the Add method does not add a new property with a name that's already shown on the Name column.

The following adds few properties with different editors, to the ExPropertiesList control:

```
With PropertiesList1
   .BeginUpdate

   .Add "ReadOnly", "", EXPROPERTIESLISTLibCtl.ReadOnly, "The property is read only"
```

```
    .Add "Edit", Me.Caption, EXPROPERTIESLISTLibCtl.Edit, "The property uses a standard edit control, to change the property's value."
    .Add "Color", Me.BackColor, EXPROPERTIESLISTLibCtl.EditColor, "The property uses a drop down color list to change the property's value."
    .Add "Font", Me.Font, EXPROPERTIESLISTLibCtl.EditFont, "The property uses a font property page to change the property's value."
    .Add "FontName", Me.Font, EXPROPERTIESLISTLibCtl.EditFontName, "The property uses a drop down font name list to change the property's value."
    .Add "Picture", Me.Icon, EXPROPERTIESLISTLibCtl.EditPicture, "The property uses picture page to change the property's value."
    .Add "Page", Me.Icon, EXPROPERTIESLISTLibCtl.EditPage, "The property uses a custom page to change the property's value."
    .Add "Boolean", Me.Visible, EXPROPERTIESLISTLibCtl.EditBoolean, "The property uses a boolean combo."
    With .Add("Enum", Me.BorderStyle, EXPROPERTIESLISTLibCtl.EditEnum, "Not available in DEMO version")
        .AddValue 0, "0 - None"
        .AddValue 1, "1 - Fixed Single"
        .AddValue 2, "2 - Sizable"
        .AddValue 3, "3 - Fixed Dialog"
        .AddValue 4, "4 - Fixed ToolWindow"
        .AddValue 5, "5 - Sizable ToolWindow"
    End With
    .Add "Date", Date, EXPROPERTIESLISTLibCtl.EditDate, "Edits a value of DATE type"
    .Add "Password", "Password", EXPROPERTIESLISTLibCtl.EditPassword, "Edits a password"
    With .Add("DropDown", "Mr.", EXPROPERTIESLISTLibCtl.EditDropDown, "Specifies a list of predefined values, but allow custom entries too.")
        .AddValue 0, "Mr."
        .AddValue 1, "Ms."
        .AddValue 2, "Dr."
    End With

    .Refresh
    .EndUpdate
End With
```

The following sample shows how to add new items for a property of EditEnum type:

```
Dim p As Property
Set p = PropertiesList1.Add("Enum", 1, EditEnum)
p.AddValue 0, "Zero"
p.AddValue 1, "One"
p.AddValue 2, "Two"
PropertiesList1.Refresh
```

You need to call Refresh method because the values for the property were unknown at adding time.

The following sample adds two COM objects to the same browser:

```
With PropertiesList1
    .BeginUpdate
      .Add "PropertiesList", PropertiesList1.Object, EXPROPERTIESLISTLibCtl.EditObject
      .Add "Form", Me, EXPROPERTIESLISTLibCtl.EditObject
    .EndUpdate
End With
```

The following sample adds a root property and two child properties:

```
With PropertiesList1
    .BeginUpdate
      .Add("Root", "", ReadOnly).ID = 1234
      .Add("Child", "", Edit, , 1234).ID = 1235
      .Add "SubChild", "", Edit, , 1235
    .EndUpdate
End With
```

property PropertiesList.AllowDrop as Boolean

Gets or sets a value indicating whether the control can accept data that the user drags into it.

Type	Description
Boolean	A Boolean expression that specifies whether the user can drag data to.

By default, the AllowDrop property is False. Currently, this property is reserved, so user should not use it.

property PropertiesList.AllowDuplicateEntries as Boolean

Specifies whether the Add method allows adding new properties with the same caption on the Name column.

Type	Description
Boolean	A Boolean expression that specifies whether the Add method allows adding new properties with the same name.

By default, the AllowDuplicateEntries property is True. If the AllowDuplicateEntries property is False, the Add method does not add a new property with a name that's already shown on the Name column.

property PropertiesList.AllowMultipleValuesOnEnum as Boolean

Specifies whether the enum types display bit combination of predefined values.

Type	Description
Boolean	A Boolean expression that indicates whether the EditEnum entries may display bit combination of predefined values. An EditEnum editor displays a list of predefined values.

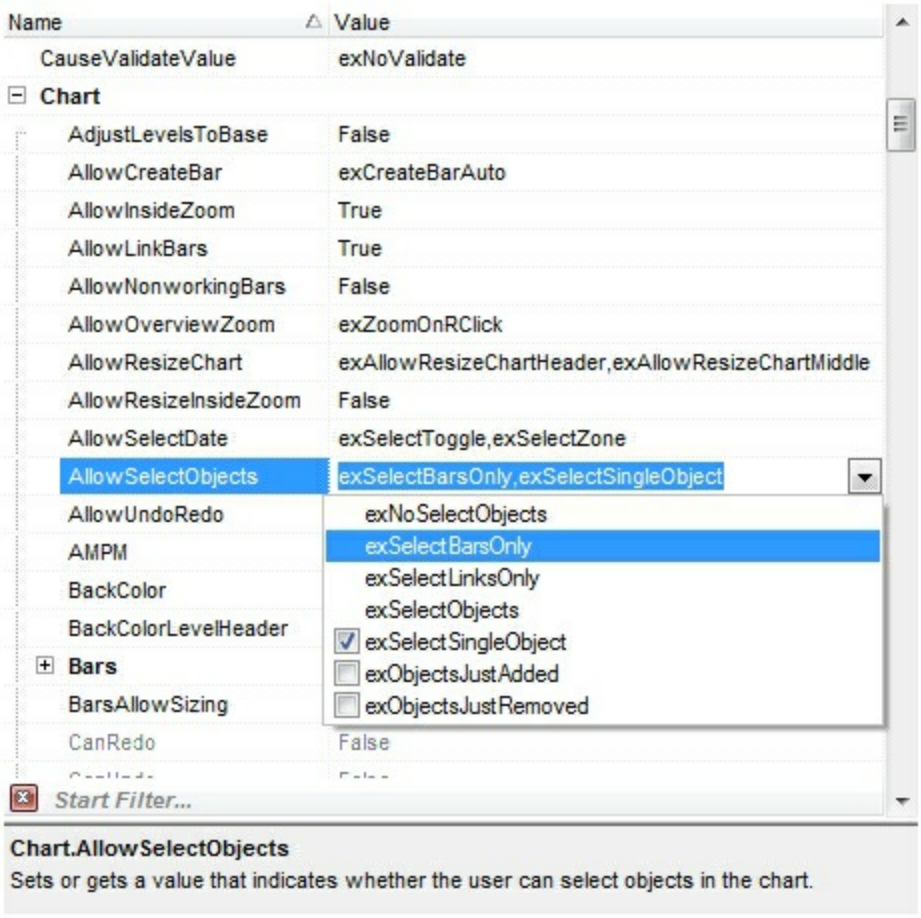
By default, the AllowMultipleValuesOnEnum property is False. If the AllowMultipleValuesOnEnum property is True, the EditEnum properties may display a checkbox for any predefined value in the enumeration that may be a bit combination. When user clicks the checkbox, the new value includes or excludes the clicked flag. The COM objects may allow bit combinations for predefined values.

For instance, let's say that our control browses the AllowSelectObjects property of the Exontrol's eXG2antt component. The AllowSelectObjects property is of SelectObjectsEnum which allow bit combination of the following values:

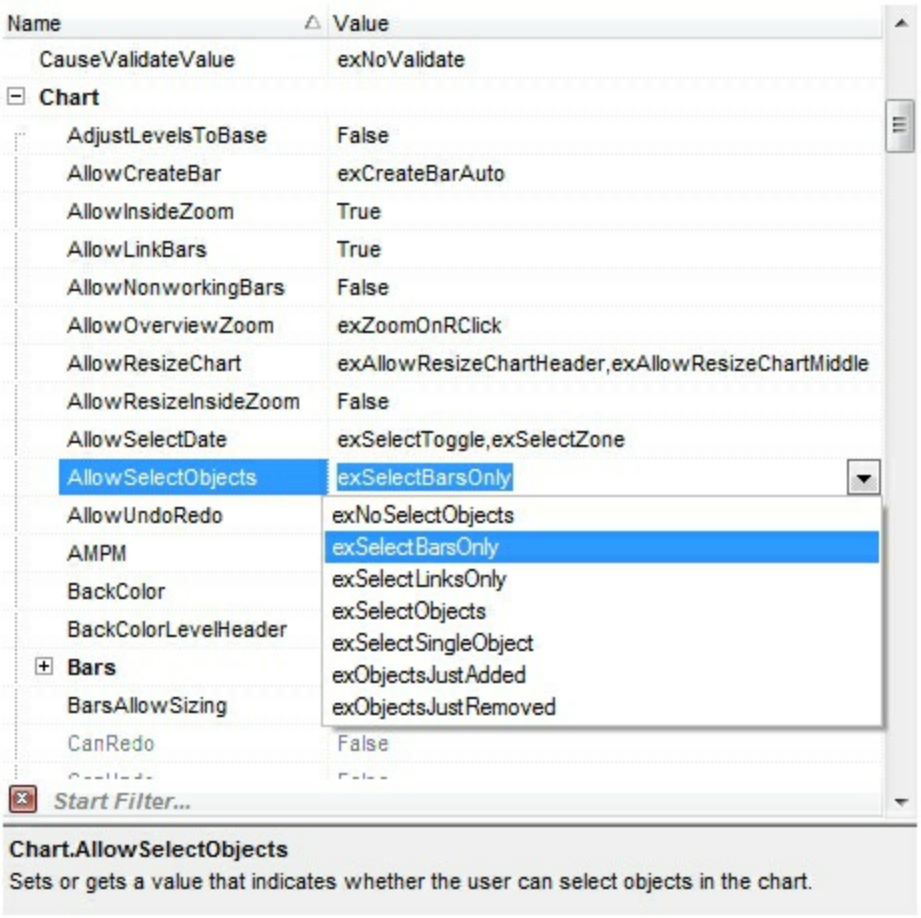
- exNoSelectObjects, 0, The user can't select any object in the chart area
- exSelectBarsOnly, 1, The user can select bars only.
- exSelectLinksOnly, 2, The user can select links only.
- exSelectObjects, 3, The user can select any object in the chart.
- **exSelectSingleObject**, 16, If present, it specifies whether the user can select one or multiple objects. For instance, the exSelectBarsOnly Or exSelectSingleObject specifies that the user can select a single bar in the chart. The exSelectLinksOnly Or exSelectSingleObject specifies that the user can select a single link in the chart.
- and so on.

The exSelectSingleObject flag can be combined with any previously value that will indicates that the control allows single selection only in the chart.

The following screen shot shows the ExPropertiesList control when the AllowMultipleValuesOnEnum property is **True**, and Chart.AllowSelectObjects property is exSelectSingleObject Or exSelectBarsOnly:



The following screen shot shows the ExPropertiesList control when the AllowMultipleValuesOnEnum property is False (by default), and Chart.AllowSelectObjects property is exSelectSingleObject Or exSelectBarsOnly.



property PropertiesList.AllowSpin as Boolean

Returns or sets a value indicating whether the control uses a spin control to edit numeric values.

Type	Description
Boolean	A boolean expression indicating whether the control uses a spin control to edit numeric values.

Use the AllowSpin property to let user changes the numeric values using a spin control. The property has effect only if the ReadOnly property is False, and it shows up only for properties of numeric type. Use the SpinStep property to hide a spin control for a specified property, or to specify the proposed change when user clicks a spin control.

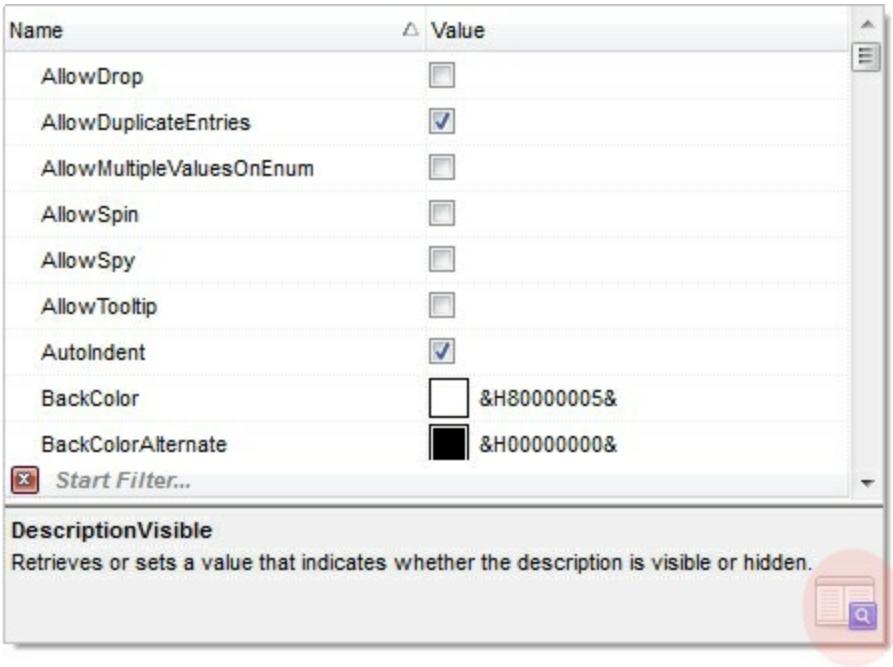
property PropertiesList.AllowSpy as Boolean

Specifies whether the control can spy other UI components or parts of them.

Type	Description
Boolean	A Boolean expression that specifies whether the control can spy other COM objects.

By default, the AllowSpy property is False. Use the AllowSpy property on True, to allow the user to browse other /COM objects at runtime by drag and drop. If the AllowSpy property is True, the control displays a spying cursor in the lower right part of the control as shown in the bellow picture. The user clicks the spy icon, the spying cursor shows up, and so it can be dragged to the object that needs to be browsed. The most part of our /COM UI components like: eXG2antt, eXGantt, eXGrid, eXTree, eXList, eXComboBox, eXPropertiesList, eXFileView, eXCalc, ... can be spied. When you do spying the component, the browser finds parts of the control that can be browsed by showing a rectangle arround the UI object. Once the user releases the button of the mouse, the properties of the object are being loaded in the current browser, and so you can change the properties of the objects at runtime.

- This movie shows how you can spy our /COM objects at runtime.



method PropertiesList.AllowSpyOn ([Handle as Variant])

Specifies handle of the window where the spy can find UI objects.

Type	Description
Handle as Variant	A long expression that specifies the handle of the window to spy.

Reserved for internal use only.

property PropertiesList.AllowTooltip as Boolean

Specifies whether the control displays a tooltip when the string value is too long.

Type	Description
Boolean	A boolean expression that indicates whether the property's tooltip is enabled or disabled.

By default, the AllowTooltip is False. If the AllowToolTip property is True, the control displays the property's tooltip if the property's name or property's value is partially visible and the cursor is over the property. Use the ToolTipDelay property to specify the time in ms that passes before the ToolTip appears. Use the ToolTip property to assign a custom tooltip to a property, that's displayed no matter if the property's name is partially visible.

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

```
Private Sub PropertiesList1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With PropertiesList1
    .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 AxPropertiesList1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXPROPERTIESLISTLib._IPropertiesListEvents_MouseMoveEvent) Handles AxPropertiesList1.MouseMoveEvent
  With AxPropertiesList1
    .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 axPropertiesList1_MouseMoveEvent(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_MouseMoveEvent e)
{
   axPropertiesList1.ShowToolTip(axPropertiesList1.get_AnchorFromPoint(-1, -1));
}
```

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

```
void OnMouseMovePropertiesList1(short Button, short Shift, long X, long Y)
{
   COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
   m_propertiesList.ShowToolTip( m_propertiesList.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 .PropertiesList1
      .ShowToolTip(.AnchorFromPoint(-1, -1))
   EndWith
endwith
```

method PropertiesList.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 PropertiesList1_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 PropertiesList.AutoDrag as AutoDragEnum

Gets or sets a value that indicates the way the component supports the AutoDrag feature.

Type	Description
AutoDragEnum	An AutoDragEnum expression that specifies what the control does once the user clicks and start dragging an item.

By default, the AutoDrag property is exAutoDragNone(0). The AutoDrag feature indicates what the control does when the user clicks an item and starts dragging it. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on. Also, you can use the AutoDrag property (exAutoDragScroll + exAutoDragScrollOnShortTouch, or 4112) to let user scrolls the control's content when user touches a capacitive screen.

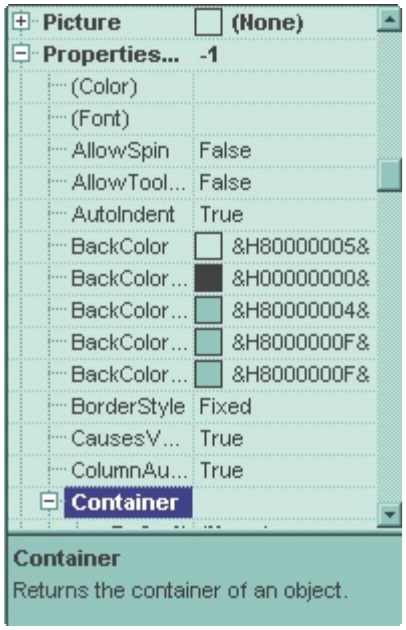
property PropertiesList.AutoIndent as Boolean

Specifies a value that indicates whether child items are automatically indented.

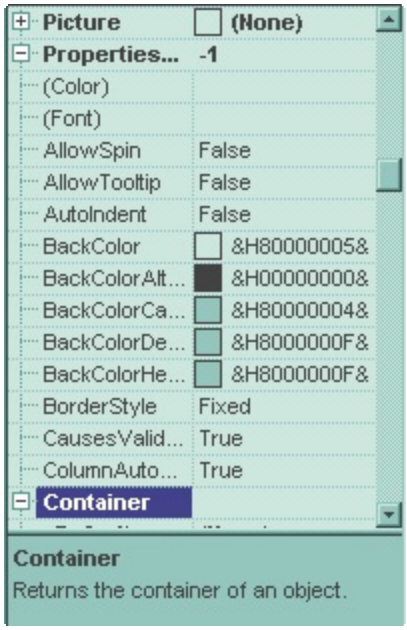
Type	Description
Boolean	A boolean expression that indicates whether the control automatically indents the child items relative to the position of their parent items.

By default, the AutoIndent property is True. Use the Indent property to specify the amount, in pixels, that child items are indented relative to their parent items. Use the HasLines property to hide the lines between parent and child items. Use HasGridLines property to draw lines between items.

Use the AutoIndent property to display all +/- signs on the first column without indentation like in the following sample.



AutoIndent = True



AutoIndent = False

property PropertiesList.BackColor as Color

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

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

Use the BackColor property to change the control's background color. Use the ForeColor property to change the foreground color. Use the BackColor property to specify the background color for a specified property. Use the BackColorAlternate property to specify the background color used to display alternate items in the control.

property PropertiesList.BackColorAlternate as Color

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

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

By default, the control's BackColorAlternate property is zero. The control ignores the BackColorAlternate property if it is 0 (zero). Use the BackColor property to specify the control's background color.

property PropertiesList.BackColorCategories as Color

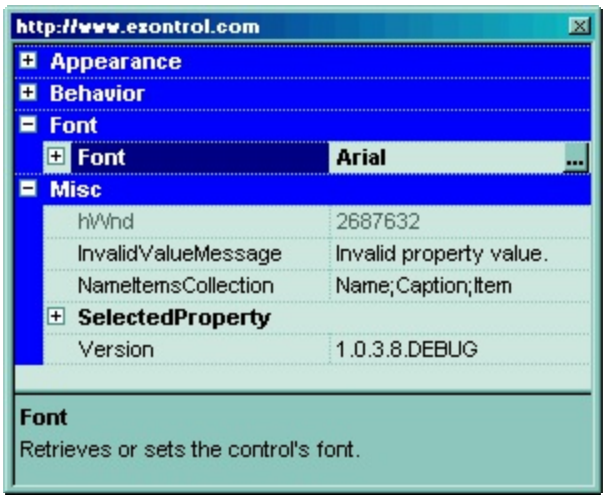
Specifies the category items background color.

Type	Description
Color	A color expression that indicates the background color for category items.

Use the ForeColorCategories and BackColorCategories properties to customize the colors for category items, when ShowCategories property is True.

The following sample displays the control's categories:

```
Private Sub Form_Load()
   With PropertiesList1
      .HasLines = False
      .BackColorCategories = vbBlue
      .ForeColorCategories = vbWhite
      .ShowCategories = True
      .ShowPropertyPages = False
      .Select PropertiesList1.Object
   End With
End Sub
```



property PropertiesList.BackColorDescription as Color

Specifies the description's background color.

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

Use the BackColorDescription property to change the the background color for control's description bar. Use the ForeColorDescription property to change the foreground color for control's description bar. Use the DescriptionVisible property to show or hide the property's description bar. Use the DescriptionHeight property to define the height of the property's description bar, in pixels.

property PropertiesList.BackColorHeader as Color

Specifies the header's background color.

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

Use the BackColorHeader and ForeColorHeader to customize colors in the control's header bar. Use the HeaderVisible property to show the control's header bar. Use the ColumnCaption property to change the column's caption.

property PropertiesList.Background(Part as BackgroundPartEnum) as Color

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

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

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

method PropertiesList.BeginUpdate ()

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

Type	Description

property PropertiesList.BorderStyle as BorderStyleEnum

Retrieves or sets the border style of the control.

Type	Description
BorderStyleEnum	A BorderStyleEnum expression that indicates the border style of the control.

Use the BorderStyle property to remove the control's border. By default, the BorderStyle property is Fixed (1).

property PropertiesList.CaptionMessageBox as String

Specifies the caption to be displayed on the message box, in case the user inputs an invalid value.

Type	Description
String	A String expression that specifies the title of the message box to be shown, when any error occurs.

By default, the CaptionMessageBox property is "Exontrol.PropertiesList". Use the CaptionMessageBox property to specify your own title when an error occurs.

method PropertiesList.Clear ()

Clears the control's content.

Type	Description

Clears all items into list. The Clear method doesn't clear the control columns collection. Use the Remove method to remove a particular property.

property PropertiesList.ColumnAutoResize as Boolean

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

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

Use the ColumnAutoResize property to fit the columns to the control's client area. By default, the ColumnAutoResize property is true. If the ColumnAutoResize property is True, the horizontal scroll bar never appears. The vertical scroll bar appears if there are items that do not fit the control's client area. Use the HeaderVisible property to display the control's header bar.

property PropertiesList.ColumnCaption([Index as Variant]) as String

Retrieves or sets the column's caption.

Type	Description
Index as Variant	A long expression that indicates the index of the column. The valid values are: 0 (Name column), 1 (Value column)
String	A string expression that indicates the column's caption.

The ColumnCaption property specifies the column's caption. Use the HeaderVisible property to show the control's header bar. The ColumnWidth property specifies the column's width. By default, the first column's caption is "Name", and the second column's caption is "Value".

property PropertiesList.ColumnsAllowSizing as Boolean

Retrieves or sets a value that indicates whether a user can resize columns at run-time.

Type	Description
Boolean	A Boolean expression that indicates whether a user can resize columns at run-time.

By default, the ColumnsAllowSizing property is False. Use the HasGridLines property to show or hide the control's grid lines. Use the HeaderVisible property to show or hide the control's header bar. Use the ColumnsAllowSizing property to resize the columns even if the control's header bar is not visible. At runtime, the user can use the CTRL + Left or CTRL + Right key to resize the columns (only if the ColumnsAllowSizing property is True).

property PropertiesList.ColumnWidth([Index as Variant]) as Long

Retrieves or sets the column's width.

Type	Description
Index as Variant	A long expression that indicates the column's index. The valid values are: 0 for "Name" column, and 1 for 'Value' column.
Long	A long expression that indicates the column's width, in pixels.

Use the ColumnWidth property to specify the column's width. Use the ColumnAutoResize property to let control resizes the columns to fit the control's client area. Use the HeaderVisible property to show the control's header bar. Use the ColumnCaption property to define the column's caption.

method PropertiesList.Copy ()

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

Type	Description

Use the Copy method to copy the control's content to the clipboard, in Enhanced Metafile (EMF) format. The Enhanced Metafile format is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following:

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

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

The Copy method copies the control's header if it's visible, and all visible items. The items are not expanded, they are listed in the order as they are displayed on the screen.

The following VB sample saves the control's content to a EMF file, when user presses the CTRL+C key:

```
Private Sub PropertiesList1_KeyDown(KeyCode As Integer, Shift As Integer)
   If (KeyCode = vbKeyC) And Shift = 2 Then
      Clipboard.Clear
      PropertiesList1.Copy
      SavePicture Clipboard.GetData(), App.Path & "\test.emf"
   End If
End Sub
```

Now, you can open your MS Windows Word application, and you can insert the file using the Insert\Picture\From File menu, or by pressing the CTRL+V key to paste the clipboard.

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

```
BOOL saveEMFtoFile( LPCTSTR szFileName )
{
   BOOL bResult = FALSE;
   if ( ::OpenClipboard( NULL ) )
```

```
  {
    CComPtr<IPicture> 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<IStream> 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 AxPropertiesList1
    .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;
axPropertiesList1.Copy();
```

property PropertiesList.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 (PropertiesList1.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 PropertiesList1.CopyTo("")
   Debug.Print i
Next
```

property PropertiesList.Count as Long

Counts the properties in the control.

Type	Description
Long	A long expression that indicates the number of items in the control.

The Count property counts the items in the control. The Count property counts the items that are on the control at one time. For instance, if you have parent items, the children items are counted as well. Use the Item property to access a Property object giving its index.

The following sample enumerates the properties in the control:

```
With PropertiesList1
   Dim i As Long
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Name
   Next
End With
```

The following sample enumerates the properties in the control using the for each statement:

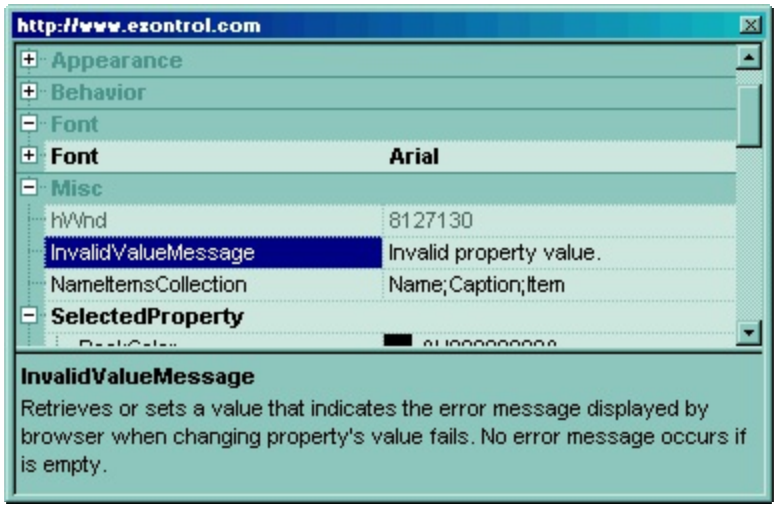
```
Dim p As EXPROPERTIESLISTLibCtl.Property
For Each p In PropertiesList1
   Debug.Print p.Name
Next
```

property PropertiesList.DefaultCategory as String

Retrieves or sets the default category.

Type	Description
String	A string expression that indicates the default category name.

Use the DefaultCategory property to specify the name of the default category. If the ShowCategories is True, The DefaultCategory category contains all properties that has no category specified (when Select method is used). By default, the DefaultCategory property is empty. The DefaultCategory property doesn't create a new category, so it should exist in the list of object categories.



property PropertiesList.DefaultItemHeight as Long

Retrieves or sets the default item height.

Type	Description
Long	A long expression that indicates the default item's height in pixels.

The DefaultItemHeight property specifies the default item's height, in pixels. By default, the DefaultItemHeight property is 16 pixels. The DefaultItemHeight property should be set before adding items. Use the Add method to add new entries to the browser. Use the Select method to load properties of a COM object.

property PropertiesList.DescriptionHeight as Long

Retrieves or sets a value that indicates the height in pixels of the description area.

Type	Description
Long	A long expression that indicates the height in pixels of the description area.

Use the DescriptionVisible property to hide the description window. The Description window displays the description for the selected property. The Description property specifies the description for the property. For instance, you can hide the description window, and you can make your own description window (in this case a Label control):

```
Private Sub PropertiesList1_SelChange()
   Label1 = PropertiesList1.SelectedProperty.Description
End Sub
```

property PropertiesList.DescriptionVisible as Boolean

Retrieves or sets a value that indicates whether the description is visible or hidden.

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

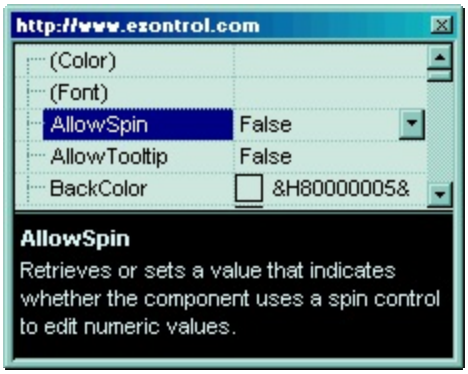
The DescriptionVisible property to hide the property's description bar. Use the DescriptionHeight property to change the description window's height. Use the BackColorDescription and ForeColorDescription properties to define the background/foreground colors for property's description bar. Use the Description property to get the property's description. Use the ToolTip property to display the property's description as tooltip.

For instance, you can hide the description window, and you can make your own description window (in this case a Label control):

```
Private Sub PropertiesList1_SelChange()
   Label1 = PropertiesList1.SelectedProperty.Description
End Sub
```

The following sample changes the property's description bar:

```
Private Sub Form_Load()
   With PropertiesList1
      .DescriptionVisible = True
      .ForeColorDescription = RGB(&HD0, &HF0, &HF0)
      .BackColorDescription = vbBlack
      .Select PropertiesList1.Object
   End With
End Sub
```



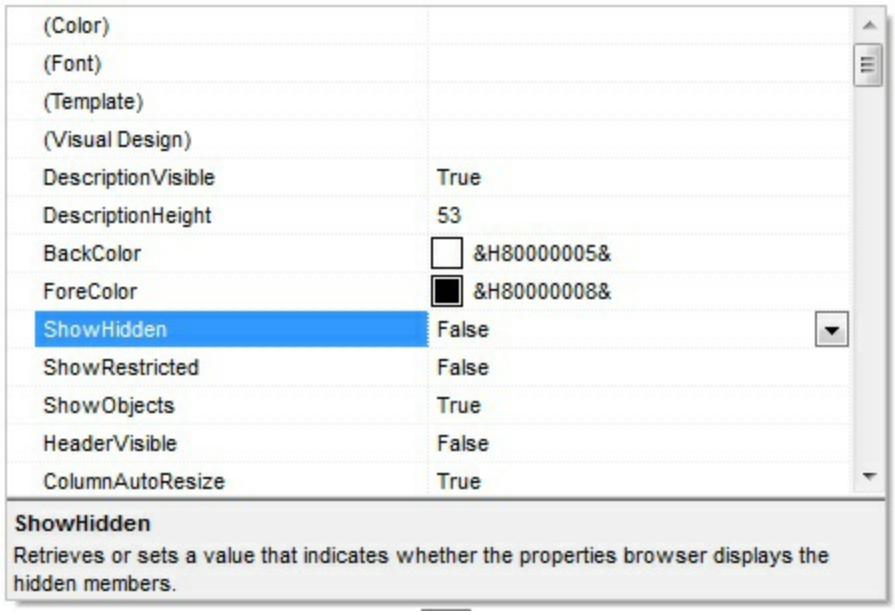
property PropertiesList.DisplayBoolAs as DisplayBoolEnum

Specifies how the properties of boolean type are displayed.

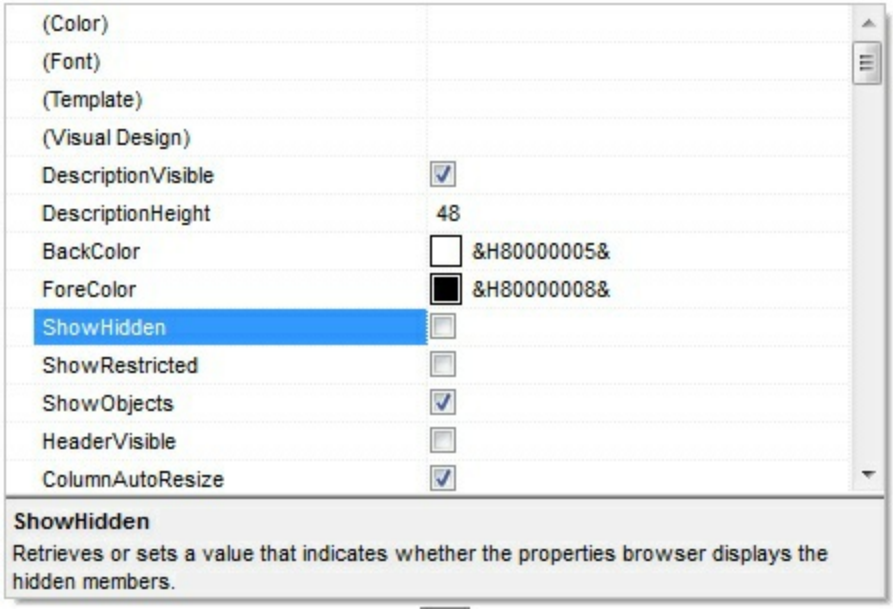
Type	Description
DisplayBoolEnum	A DisplayBoolEnum expression that specifies the way the control displays the boolean properties.

By default, the DisplayBoolAs property is exBoolEnum, so the True and False values are displayed for boolean properties. Use the DisplayBoolAs property on exBoolCheck to display the boolean properties using the check-boxes.

The following screen shot shows the boolean properties using the exBoolEnum (by default):



The following screen shot shows the boolean properties using the exBoolCheck:



property PropertiesList.DisplayColorAs as DisplayColorEnum

Specifies how the properties of color type are displayed.

Type	Description
DisplayColorEnum	A DisplayColorEnum expression that indicates how the properties of color type are displayed.

By default, all color properties are displayed using the &HXXXXXXXX& form, where X is a hex value. Use the DisplayColorAs property to change how the properties display the color values.

The following screen shot shows the colors using the exDefault (by default):

The following screen shot shows the colors using the exRGB:

property PropertiesList.EditOnKey as Long

Customizes the F4 key to let user edits a property using the keys.

Type	Description
Long	A long expression that indicates the key code used instead F4 key.

If the EditOnSelect property is False, you can start editing a property using the keyboard using the F4 key. Use the EditOnKey property to specify the key being used to open the property's editor when EditOnSelect property is False. By default, the EditOnKey property is VK_F4.

property PropertiesList.EditOnSelect as Boolean

Retrieves or sets a value that indicates whether the properties browser is ready to edit a value when the selection is changed.

Type	Description
Boolean	A boolean expression that indicates whether the properties browser is ready to edit a value when the selection is changed.

By default, EditOnSelect property is False. Use the ReadOnly property to avoid editing the properties.

property PropertiesList.Enabled as Boolean

Enables or disables the control.

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

Use the Enabled property to disable the control. Use the ReadOnly property to disable editing properties. A disabled control looks grayed. Use the Locked property to lock a property. Use the Enabled property to disable a property.

method PropertiesList.EndUpdate ()

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

Type	Description

This method prevents the control from painting until the EndUpdate method is called. The BeginUpdate and EndUpdate methods increases the speed of loading your custom properties , by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too.

property PropertiesList.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 PropertiesList.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 PropertiesList1.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*

method PropertiesList.ExpandAll ()

Expands all items.

Type	Description

The ExpandAll method expands all items. Use the ExpandItem method to expand an item. Use the ExpandOnSearch property to expand items automatically while user types characters to search for a specific property.

property PropertiesList.ExpandItem(Name as Variant) as Boolean

Expands or collapses an item.

Type	Description
Name as Variant	A string expression that indicates the property's name, or a long expression that indicates the property's identifier.
Boolean	A boolean expression that indicates whether the item is expanded or collapsed. True means that the item is expanded, and False means that the item is collapsed.

Use the ExpandItem property to expand or collapse an item. Use the ID property to determine the property's identifier. Use the Name property to retrieve the property's name. Use the ExpandAll method to expand all items. Use the ExpandOnSearch property to expand items automatically while user types characters to search for a specific property.

The following samples collapses the "Appearance" item:

```
Private Type POINTAPI
      x As Long
      y As Long
End Type

Private Type MSG
   hwnd As Long
   message As Long
   wParam As Long
   lParam As Long
   time As Long
   pt As POINTAPI
End Type
Private Declare Function PeekMessage Lib "user32" Alias "PeekMessageA" (lpMsg As MSG,
ByVal hwnd As Long, ByVal wMsgFilterMin As Long, ByVal wMsgFilterMax As Long, ByVal
wRemoveMsg As Long) As Long
Private Const PM_NOREMOVE = &H0
Private Declare Function TranslateMessage Lib "user32" (lpMsg As MSG) As Long
Private Declare Function DispatchMessage Lib "user32" Alias "DispatchMessageA" (lpMsg
As MSG) As Long
```

```
Private Sub Form_Load()
    With PropertiesList1
        .BeginUpdate
            .HasLines = False
            .ShowCategories = True
            .MarkCategories = True
            .Select PropertiesList1.Object

            ' When using the Select method, the list of properties is not immediately available, so we need to proceeds few messages
            Dim m As MSG
            While PeekMessage(m, .hwnd, 0, 0, 1)
                TranslateMessage m
                DispatchMessage m
            Wend

            .ExpandItem("Appearance") = False
        .EndUpdate
    End With
End Sub
```

property PropertiesList.ExpandOnSearch as Boolean

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

Type	Description
Boolean	A Boolean expression that indicates whether the control automatically expands items with children items when the user searches for typed characters (incremental search)

By default, the ExpandOnSearch property is False. The IncrementalSearch property specifies the control's incremental searching type. If the ExpandOnSearch property is True, and user starts typing characters within the control, it takes each item, if the item is not found but it has child items, it expands the item, and start looking inside the child items, and so on. The property has effect only if the items display child items. When using the ExpandOnSearch property you have to be carefully that your browsed object does not provide recursive objects. In other words, an object that returns an already browsed object and so on. Use the ExpandItem property to programmatically expand/collapse an item. Use the ExpandAll method to expand all items.

property PropertiesList.FilterBarFont as IFontDisp

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

Type	Description
IFontDisp	A font object that indicates the font used to paint the description for control's filter

Use the FilterBarFont property to specify the font for the control's filter bar object. Use the Font property to set the control's font. Use Use the Refresh method to refresh the control.

property PropertiesList.FilterBarPrompt as String

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

Type	Description
String	A string expression that indicates the HTML caption being displayed in the filter bar, when filter prompt pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature.

By default, the FilterBarPrompt property is "<fgcolor=808080>Start Filter...</fgcolor>". The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature. Changing the FilterBarPrompt property won't change the current filter.

The FilterBarPrompt property has effect only if:

- FilterBarPromptVisible property is True
- FilterBarPromptPattern property is Empty.

The FilterBarPrompt property supports HTML format as described here:

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

line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

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

blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

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

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

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

property PropertiesList.FilterBarPromptPattern as String

Specifies the pattern for the filter prompt.

Type	Description
String	A string expression that specifies the pattern to filter the list.

By default, the FilterBarPromptPattern property is empty. If the FilterBarPromptPattern property is empty, the filter bar displays the FilterBarPrompt property, if the FilterBarPromptVisible property is True. The FilterBarPromptPattern property indicates the patter to filter the list. The IncrementalSearch property indicates the type of filtering.

The filter prompt works based on the IncrementalSearch property as follows:

- exStartWith, the list contains only items that starts with filter prompt's text (FilterBarPromptPattern property).
- exContains, the list displays only items that contains the filter prompt's text (FilterBarPromptPattern property).

property PropertiesList.FilterBarPromptVisible as FilterBarVisibleEnum

Shows or hides the filter prompt.

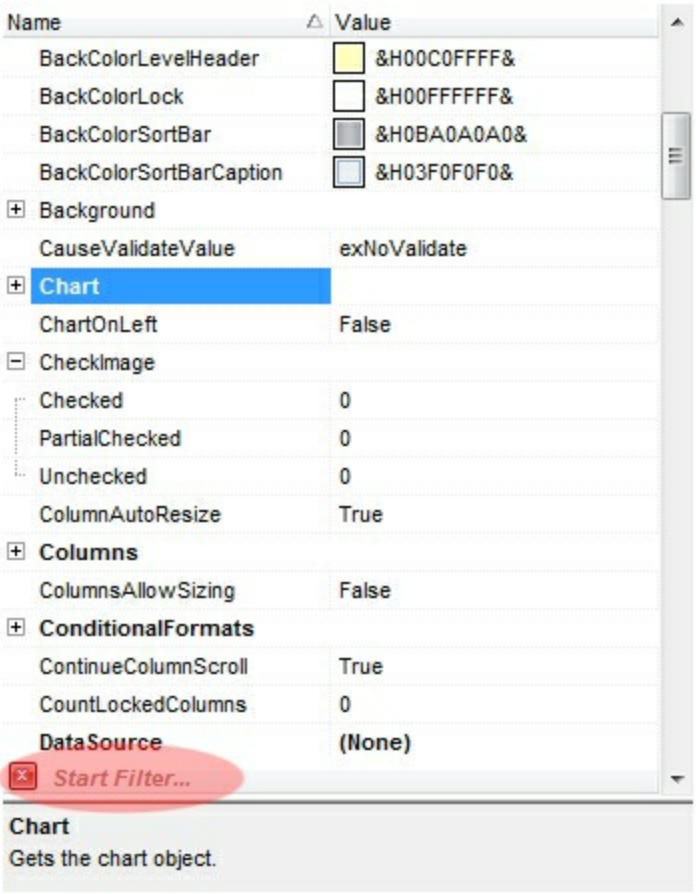
Type	Description
FilterBarVisibleEnum	A FilterBarVisibleEnum expression that specifies whether the control's filter prompt is visible or hidden.

By default, the FilterBarPromptVisible property is False, in other words the filter prompt is not visible. Use the FilterBarPromptVisible property to add filtering capabilities to the component. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. The IncrementalSearch property indicates the type of filtering.

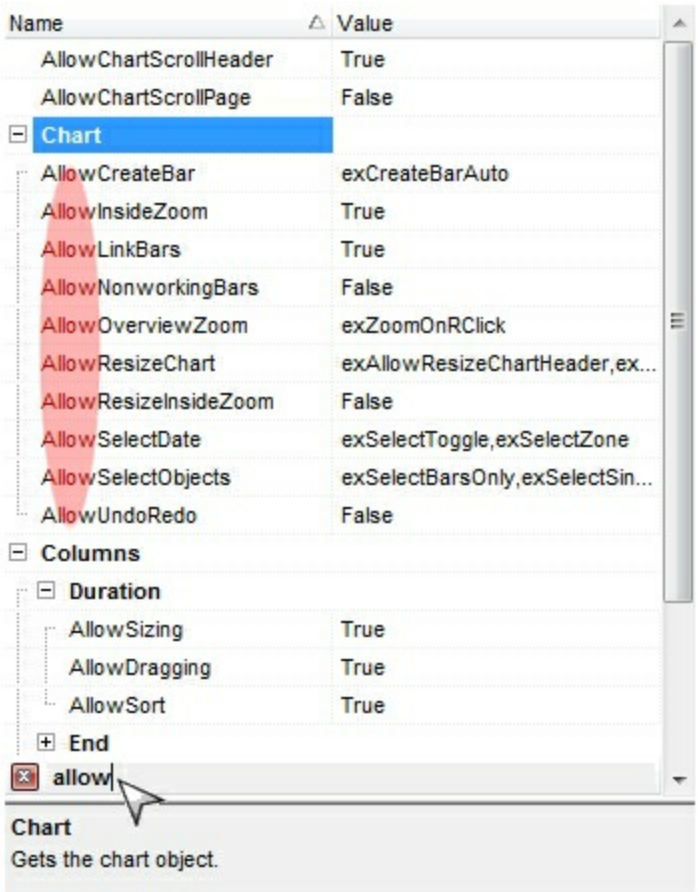
The filter prompt works based on the IncrementalSearch property as follows:

- exStartWith, the list contains only items that starts with filter prompt's text (FilterBarPromptPattern property).
- exContains, the list displays only items that contains the filter prompt's text (FilterBarPromptPattern property).

The following screen shot shows the filter prompt (FilterBarPromptVisible property is True):



The following screen shot shows the list once the user types "allow" in the filter prompt portion:



property PropertiesList.FireIncludeProperty as Boolean

Retrieves or sets a value that indicates whether the IncludeProperty event is fired.

Type	Description
Boolean	A Boolean expression that indicates whether the IncludeProperty event is fired.

By default, the FireIncludeProperty is True. Use the FireIncludeProperty to allow the control filtering properties using the IncludeProperty event. If the FireIncludeProperty is False, the control doesn't fire the IncludeProperty event. Use the FireIncludeProperty property on False, when your properties browser doesn't require custom filtering, to increase the speed of loading objects into the browser.

The following sample excludes the "hPal" variable of a Picture property:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   If Property.Variable = True Then
      If Property.Name = "hPal" Then
         Cancel = True
      End If
   End If
End Sub
```

property PropertiesList.Font as IFontDisp

Retrieves or sets the control's font.

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

Use the Font property to change the font object used to paint the items. An object property appears as bolded.

property PropertiesList.ForeColor as Color

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

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

The ForeColor property determines the foregound color used to paint the items. Use BackColor property to change the control's background color.

property PropertiesList.ForeColorCategories as Color

Specifies the category items foreground color.

Type	Description
Color	A color expression that indicates the category items foreground color..

Use the ForeColorCategories and BackColorCategories properties to customize the color for category items. The ForeColorCategories property has effect only if the ShowCategories property is True. Use the ForeColor property to define the control's foreground color.

The following sample displays the control's categories:

```
Private Sub Form_Load()
   With PropertiesList1
      .HasLines = False
      .BackColorCategories = vbBlue
      .ForeColorCategories = vbWhite
      .ShowCategories = True
      .ShowPropertyPages = False
      .Select PropertiesList1.Object
   End With
End Sub
```

property PropertiesList.ForeColorDescription as Color

Specifies the description's foreground color.

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

Use the ForeColorDescription property to change the the foreground color for control's description bar. Use the BackColorDescription property to change the background color for control's description bar. Use the DescriptionVisible property to show or hide the property's description bar. Use the DescriptionHeight property to define the height of the property's description bar, in pixels.

property PropertiesList.ForeColorHeader as Color

Specifies the header's foreground color.

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

Use the BackColorHeader and ForeColorHeader to customize colors in the control's header bar. Use the HeaderVisible property to show the control's header. bar.

property PropertiesList.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 PropertiesList.GridLineColor as Color

Retrieves or sets the grid line color.

Type	Description
Color	A color expression that indicates the grid line color.

Use the GridLineColor property to specify the control's grid line color. The HasLines property tnhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their parent items.

property PropertiesList.HasButtons as ExpandButtonEnum

Adds a button to the left side of each parent item. The user can click the button to expand or collapse the child items as an alternative to double-clicking the parent item.

Type	Description
ExpandButtonEnum	An ExpandButtonEnum expression that indicates whether the left side button of each parent item is visible or hidden.

By default, the HasButtons property is exPlus. The HasButtons property defines the visual appearance for the expanding/collapsing buttons. The HasButtonsCustom property specifies the index of icons being used for +/- signs on parent items, when HasButtons property is exCustom.

The following VB sample changes the +/- button appearance:

```
With PropertiesList1
   .HasButtons = ExpandButtonEnum.exWPlus
End With
```

The following C++ sample changes the +/- button appearance:

```
m_PropertiesList.SetHasButtons( 3 /*exWPlus*/ );
```

The following VB.NET sample changes the +/- button appearance:

```
With AxPropertiesList1
   .HasButtons = EXPROPERTIESLISTLib.ExpandButtonEnum.exWPlus
End With
```

The following C# sample changes the +/- button appearance:

```
axPropertiesList1.HasButtons = EXPROPERTIESLISTLib.ExpandButtonEnum.exWPlus;
```

The following VFP sample changes the +/- button appearance:

```
with thisform.PropertiesList1
   .HasButtons = 3 && exWPlus
endwith
```

property PropertiesList.HasButtonsCustom(Expanded as Boolean) as Long

Specifies the index of icons for +/- signs when the HasButtons property is exCustom.

Type	Description
Expanded as Boolean	A boolean expression that indicates the sign being changed.
Long	A long expression that indicates the icon being used for +/- signs on the parent items. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object.

Use the HasButtonsCustom property to assign custom icons to the +/- signs on the parent items. The HasButtonsCustom property has effect only if the HasButtons property is exCustom. Use the Images, ReplaceIcon methods to add new icons to the control, at runtime. Use the HTMLPicture property to display icons in the node's caption.

The following VB sample specifies different (as in the screen shot) +/- signs for the control:

```
With PropertiesList1
   .BeginUpdate
      .Images
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExm

      .HasGridLines = exNoLines
      .HasButtons = exCustom
      .HasButtonsCustom(False) = 1
      .HasButtonsCustom(True) = 2
      .HTMLPicture("S1") =
"gAAAABgYACEHgUJFEEAAWhUJCEJEEJggEhMCYEXjUbjkJQECj8gj8hAEjkshQEpADAlkJf8C

      With .Add("I1", "", ReadOnly)
         .HTMLName = "S1 Item 1"
      End With
      With .Add("Subitem 1", "", Edit, , "I1")
      End With
```

```
    With .Add("Subitem 1.1", "", Edit, , "Subitem 1")
    End With
    With .Add("Subitem 2", "", Edit, , "I1")
    End With
  .EndUpdate
End With
```

property PropertiesList.HasGridLines as GridLinesEnum

Retrieves or sets a value that indicates whether the grid lines are visible or hidden.

Type	Description
GridLinesEnum	A GridLinesEnum expression that indicates whether the grid lines are visible or hidden.

Use the HasGridLine property to show or hide the grid lines. The grid lines are painted to mark columns and items. Use the HasLines to draw the lines that link child items to their parent items. Use HasButtons property to hide the buttons displayed at the left of each parent item.

property PropertiesList.HasLines as Boolean

Enhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their parent items.

Type	Description
Boolean	A boolean expression that indicates whether the control draws the lines that link child items to their parents.

Use the HasGridLines property to draw the grid lines. Use the Indent property to specify the amount, in pixels, that child items are indented relative to their parent items. Use the AutoIndent property to indicate whether child items are automatically indented.

property PropertiesList.HeaderAppearance as AppearanceEnum

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

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

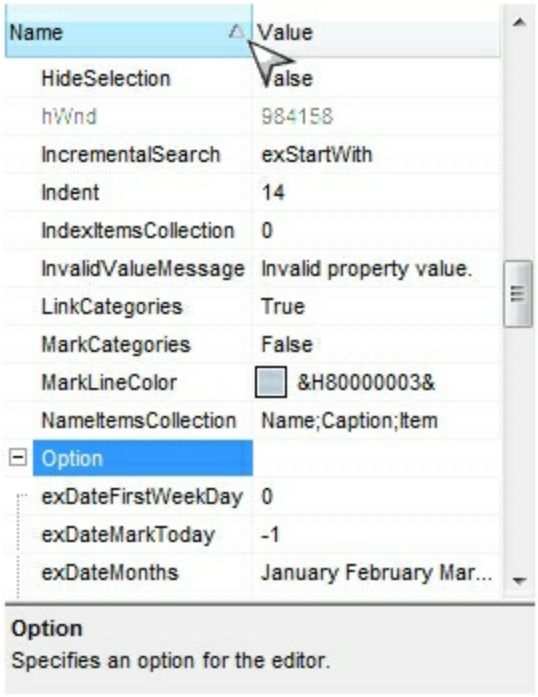
Use the HeaderAppearance property to change the appearance of the control's header bar. Use the HeaderVisible property to hide the control's header bar. Use the BackColorHeader and ForeColorHeader properties to define the background/foreground colors for the control's header bar.

property PropertiesList.HeaderHeight as Long

Retrieves or sets a value indicating the control's header height.

Type	Description
Long	A Long expression that specifies the height of the control's header (in pixels).

By default, the height of the header bar is 18 pixels. The HeaderHeight property has effect while the control's header bar is visible. The user can use the control's header bar to sort or order the visible columns. The HeaderVisible property indicates whether the control's header bar is visible or hidden. Use the BackColorHeader and ForeColorHeader properties to define the background/foreground colors for the control's header bar. Use the ColumnCaption property to change the column's caption.

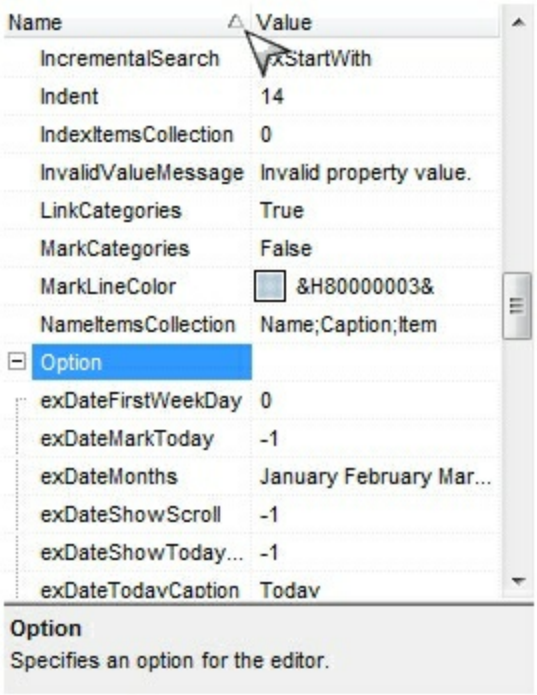


property PropertiesList.HeaderVisible as Boolean

Retrieves or sets a value that indicates whether the control's header is visible or hidden.

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

By default, the control's header bar is hidden. The user can use the control's header bar to sort or order the visible columns. If the HeaderVisible property is True, the control shows its header bar. If the header bar is visible, the user is able to resize the columns by dragging the mouse. For instance, you can show the control's header and set the ColumnAutoResize property to False to let the user be able to resize the columns at runtime. Use the BackColorHeader and ForeColorHeader properties to define the background/foreground colors for the control's header bar. Use the ColumnCaption property to change the column's caption. Use the ColumnsAllowSizing property to specify whether the user can resize the columns at run-time, even if the control's header bar is hidden. The HeaderHeight property specifies the height of the control's header bar. Use the HeaderAppearance property to change the appearance of the control's header bar.



The following VB6 sample displays the control's header bar:

```
Private Sub Form_Load()
  With PropertiesList1
    .BeginUpdate
      .HeaderVisible = True
      .ColumnCaption(0) = "Property"
```

```
        .Select .Object
    .EndUpdate
  End With
End Sub
```

property PropertiesList.HideSelection as Boolean

Specifies whether selected property appears selected when the control loses focus.

Type	Description
Boolean	A boolean expression that indicates whether selected property appears selected when the control loses focus.

Use the HideSelection property to hide the selection when the control loses the focus. By default, the HideSelection property is False.

property PropertiesList.HotBackColor as Color

Retrieves or sets a value that indicates the hot-tracking background color.

Type	Description
Color	A color expression that indicates the background color for item from the cursor (hovering the item). 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 HotBackColor property is 0, which means that the HotBackColor property has no effect. Use the HotBackColor property on a non-zero value to highlight the item from the cursor. The HotForeColor property specifies the foreground color to highlight the item from the cursor. The SelBackColor property specifies the selection background color.

The following sample displays a different background color mouse passes over an item.

VBA (MS Access, Excell...)

```
With PropertiesList1
   .BeginUpdate
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   .Select PropertiesList1
   .EndUpdate
End With
```

VB6

```
With PropertiesList1
   .BeginUpdate
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   .Select PropertiesList1
   .EndUpdate
End With
```

VB.NET

```
With Expropertieslist1
   .BeginUpdate()
   .HotBackColor = Color.FromArgb(0,0,128)
   .HotForeColor = Color.FromArgb(255,255,255)
   .Select(Expropertieslist1)
   .EndUpdate()
End With
```

VB.NET for /COM

```
With AxPropertiesList1
   .BeginUpdate()
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   .Select(AxPropertiesList1.GetOcx())
   .EndUpdate()
End With
```

C++

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

   #import <ExPropertiesList.dll>
   using namespace EXPROPERTIESLISTLib;
*/
EXPROPERTIESLISTLib::IPropertiesListPtr spPropertiesList1 =
GetDlgItem(IDC_PROPERTIESLIST1)->GetControlUnknown();
spPropertiesList1->BeginUpdate();
spPropertiesList1->PutHotBackColor(RGB(0,0,128));
spPropertiesList1->PutHotForeColor(RGB(255,255,255));
spPropertiesList1->Select(spPropertiesList1);
spPropertiesList1->EndUpdate();
```

C++ Builder

```
PropertiesList1->BeginUpdate();
PropertiesList1->HotBackColor = RGB(0,0,128);
PropertiesList1->HotForeColor = RGB(255,255,255);
PropertiesList1->Select(PropertiesList1);
PropertiesList1->EndUpdate();
```

C#

```
expropertieslist1.BeginUpdate();
expropertieslist1.HotBackColor = Color.FromArgb(0,0,128);
expropertieslist1.HotForeColor = Color.FromArgb(255,255,255);
expropertieslist1.Select(expropertieslist1);
expropertieslist1.EndUpdate();
```

JavaScript

```
<OBJECT classid="clsid:A703DF80-DFF3-48D7-A4C7-47CF6A48425C"
id="PropertiesList1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
   PropertiesList1.BeginUpdate();
   PropertiesList1.HotBackColor = 8388608;
   PropertiesList1.HotForeColor = 16777215;
   PropertiesList1.Select(PropertiesList1);
   PropertiesList1.EndUpdate();
</SCRIPT>
```

C# for /COM

```
axPropertiesList1.BeginUpdate();
axPropertiesList1.HotBackColor = Color.FromArgb(0,0,128);
axPropertiesList1.HotForeColor = Color.FromArgb(255,255,255);
axPropertiesList1.Select(axPropertiesList1.GetOcx());
axPropertiesList1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   ;

   super();

   expropertieslist1.BeginUpdate();
   expropertieslist1.HotBackColor(WinApi::RGB2int(0,0,128));
   expropertieslist1.HotForeColor(WinApi::RGB2int(255,255,255));
   expropertieslist1.Select(expropertieslist1);
   expropertieslist1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxPropertiesList1 do
begin
   BeginUpdate();
   HotBackColor := Color.FromArgb(0,0,128);
   HotForeColor := Color.FromArgb(255,255,255);
   Select(AxPropertiesList1);
   EndUpdate();
end
```

Delphi (standard)

```
with PropertiesList1 do
begin
   BeginUpdate();
   HotBackColor := RGB(0,0,128);
   HotForeColor := RGB(255,255,255);
   Select(PropertiesList1);
   EndUpdate();
end
```

VFP

```
with thisform.PropertiesList1
```

```
   .BeginUpdate
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   .Select(thisform.PropertiesList1)
   .EndUpdate
endwith
```

dBASE Plus

```
local oPropertiesList

oPropertiesList = form.Activex1.nativeObject
oPropertiesList.BeginUpdate()
oPropertiesList.HotBackColor = 0x800000
oPropertiesList.HotForeColor = 0xffffff
oPropertiesList.Select(oPropertiesList)
oPropertiesList.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oPropertiesList as P

oPropertiesList = topparent:CONTROL_ACTIVEX1.activex
oPropertiesList.BeginUpdate()
oPropertiesList.HotBackColor = 8388608
oPropertiesList.HotForeColor = 16777215
oPropertiesList.Select(oPropertiesList)
oPropertiesList.EndUpdate()
```

Visual Objects

```

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:HotBackColor := RGB(0,0,128)
oDCOCX_Exontrol1:HotForeColor := RGB(255,255,255)
oDCOCX_Exontrol1:Select(oDCOCX_Exontrol1)
```

```
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oPropertiesList

oPropertiesList = ole_1.Object
oPropertiesList.BeginUpdate()
oPropertiesList.HotBackColor = RGB(0,0,128)
oPropertiesList.HotForeColor = RGB(255,255,255)
oPropertiesList.Select(oPropertiesList)
oPropertiesList.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComHotBackColor to (RGB(0,0,128))
   Set ComHotForeColor to (RGB(255,255,255))
   Send ComSelect (pvComObject(Self))
   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 oPropertiesList

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

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

   oPropertiesList := XbpActiveXControl():new( oForm:drawingArea )
   oPropertiesList:CLSID  := "Exontrol.PropertiesList.1" /*{A703DF80-DFF3-48D7-
A4C7-47CF6A48425C}*/
   oPropertiesList:create(,, {10,60},{610,370} )

      oPropertiesList:BeginUpdate()
      oPropertiesList:SetProperty("HotBackColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 0,0,128 } )  , .F. ))
      oPropertiesList:SetProperty("HotForeColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 255,255,255 } )  , .F. ))
      oPropertiesList:Select(oPropertiesList)
      oPropertiesList:EndUpdate()

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

property PropertiesList.HotForeColor as Color

Retrieves or sets a value that indicates the hot-tracking foreground color.

Type	Description
Color	A color expression that indicates the foreground color for item from the cursor (hovering the item).

By default, the HotForeColor property is 0, which means that the HotForeColor property has no effect. Use the HotForeColor property on a non-zero value to highlight the item from the cursor. The HotBackColor property specifies the background color to highlight the item from the cursor. The SelForeColor property specifies the selection foreground color.

The following sample displays a different background color mouse passes over an item.

VBA (MS Access, Excell...)

```
With PropertiesList1
   .BeginUpdate
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   .Select PropertiesList1
   .EndUpdate
End With
```

VB6

```
With PropertiesList1
   .BeginUpdate
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   .Select PropertiesList1
   .EndUpdate
End With
```

VB.NET

```
With Expropertieslist1
   .BeginUpdate()
   .HotBackColor = Color.FromArgb(0,0,128)
   .HotForeColor = Color.FromArgb(255,255,255)
```

```
    .Select(Expropertieslist1)
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxPropertiesList1
    .BeginUpdate()
    .HotBackColor = RGB(0,0,128)
    .HotForeColor = RGB(255,255,255)
    .Select(AxPropertiesList1.GetOcx())
    .EndUpdate()
End With
```

C++

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

    #import <ExPropertiesList.dll>
    using namespace EXPROPERTIESLISTLib;
*/
EXPROPERTIESLISTLib::IPropertiesListPtr spPropertiesList1 =
GetDlgItem(IDC_PROPERTIESLIST1)->GetControlUnknown();
spPropertiesList1->BeginUpdate();
spPropertiesList1->PutHotBackColor(RGB(0,0,128));
spPropertiesList1->PutHotForeColor(RGB(255,255,255));
spPropertiesList1->Select(spPropertiesList1);
spPropertiesList1->EndUpdate();
```

C++ Builder

```
PropertiesList1->BeginUpdate();
PropertiesList1->HotBackColor = RGB(0,0,128);
PropertiesList1->HotForeColor = RGB(255,255,255);
```

```
PropertiesList1->Select(PropertiesList1);
PropertiesList1->EndUpdate();
```

C#

```
expropertieslist1.BeginUpdate();
expropertieslist1.HotBackColor = Color.FromArgb(0,0,128);
expropertieslist1.HotForeColor = Color.FromArgb(255,255,255);
expropertieslist1.Select(expropertieslist1);
expropertieslist1.EndUpdate();
```

JavaScript

```
<OBJECT classid="clsid:A703DF80-DFF3-48D7-A4C7-47CF6A48425C"
id="PropertiesList1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
   PropertiesList1.BeginUpdate();
   PropertiesList1.HotBackColor = 8388608;
   PropertiesList1.HotForeColor = 16777215;
   PropertiesList1.Select(PropertiesList1);
   PropertiesList1.EndUpdate();
</SCRIPT>
```

C# for /COM

```
axPropertiesList1.BeginUpdate();
axPropertiesList1.HotBackColor = Color.FromArgb(0,0,128);
axPropertiesList1.HotForeColor = Color.FromArgb(255,255,255);
axPropertiesList1.Select(axPropertiesList1.GetOcx());
axPropertiesList1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
```

```
    ;

    super();

    expropertieslist1.BeginUpdate();
    expropertieslist1.HotBackColor(WinApi::RGB2int(0,0,128));
    expropertieslist1.HotForeColor(WinApi::RGB2int(255,255,255));
    expropertieslist1.Select(expropertieslist1);
    expropertieslist1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxPropertiesList1 do
begin
    BeginUpdate();
    HotBackColor := Color.FromArgb(0,0,128);
    HotForeColor := Color.FromArgb(255,255,255);
    Select(AxPropertiesList1);
    EndUpdate();
end
```

Delphi (standard)

```
with PropertiesList1 do
begin
    BeginUpdate();
    HotBackColor := RGB(0,0,128);
    HotForeColor := RGB(255,255,255);
    Select(PropertiesList1);
    EndUpdate();
end
```

VFP

```
with thisform.PropertiesList1
    .BeginUpdate
    .HotBackColor = RGB(0,0,128)
```

```
   .HotForeColor = RGB(255,255,255)
   .Select(thisform.PropertiesList1)
   .EndUpdate
endwith
```

dBASE Plus

```
local oPropertiesList

oPropertiesList = form.Activex1.nativeObject
oPropertiesList.BeginUpdate()
oPropertiesList.HotBackColor = 0x800000
oPropertiesList.HotForeColor = 0xffffff
oPropertiesList.Select(oPropertiesList)
oPropertiesList.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oPropertiesList as P

oPropertiesList = topparent:CONTROL_ACTIVEX1.activex
oPropertiesList.BeginUpdate()
oPropertiesList.HotBackColor = 8388608
oPropertiesList.HotForeColor = 16777215
oPropertiesList.Select(oPropertiesList)
oPropertiesList.EndUpdate()
```

Visual Objects

```
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:HotBackColor := RGB(0,0,128)
oDCOCX_Exontrol1:HotForeColor := RGB(255,255,255)
oDCOCX_Exontrol1:Select(oDCOCX_Exontrol1)
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oPropertiesList

oPropertiesList = ole_1.Object
oPropertiesList.BeginUpdate()
oPropertiesList.HotBackColor = RGB(0,0,128)
oPropertiesList.HotForeColor = RGB(255,255,255)
oPropertiesList.Select(oPropertiesList)
oPropertiesList.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComHotBackColor to (RGB(0,0,128))
   Set ComHotForeColor to (RGB(255,255,255))
   Send ComSelect (pvComObject(Self))
   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 oPropertiesList

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

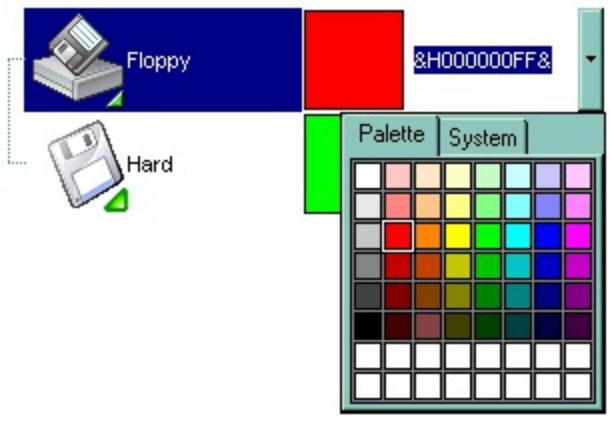
```
   oPropertiesList := XbpActiveXControl():new( oForm:drawingArea )
   oPropertiesList:CLSID  := "Exontrol.PropertiesList.1" /*{A703DF80-DFF3-48D7-
A4C7-47CF6A48425C}*/
   oPropertiesList:create(,, {10,60},{610,370} )

      oPropertiesList:BeginUpdate()
      oPropertiesList:SetProperty("HotBackColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 0,0,128 } )  , .F. ))
      oPropertiesList:SetProperty("HotForeColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 255,255,255 } )  , .F. ))
      oPropertiesList:Select(oPropertiesList)
      oPropertiesList:EndUpdate()

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

property PropertiesList.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). Use the HTMLName property to display HTML format in the Name column.



The following template sample adds two properties and a custom size picture for each property:

```
BeginUpdate
HasGridLines = False
DefaultItemHeight = 52
HTMLPicture("floppy") = "D:\Temp\Icons\3floppy_1mount.gif"
HTMLPicture("hard") = "D:\Temp\Icons\3floppy_mount.gif"
Add("Floppy", "", EditColor).HTMLName = "<img>floppy</img>Floppy"
Add("Hard", "", EditColor).HTMLName = "<img>hard</img>Hard"
EndUpdate
```

property PropertiesList.hWnd as Long

Retrieves the control's window handle.

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

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

method PropertiesList.Images (Handle as Variant)

Sets the control's image list at runtime.

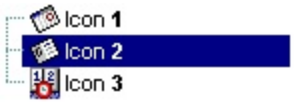
Type	Description
Handle as Variant	The Handle parameter can be: • A string expression that specifies the ICO file to add. The ICO file format is an image file format for computer icons in Microsoft Windows. ICO files contain one or more small images at multiple sizes and color depths, such that they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

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

Use the Images method to attach a list of icons to the control. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. At runtime, the user can use the Images and ReplaceIcon method to change the Images collection. Use the HTMLName property to assign built-in HTML format to a property. In design mode, user can add icons to the control using the control's Template page. No matter what programming language you are using, you can have a quick view of the component's features using the **WYSWYG Template** editor. It's a nice feature and we don't want you to miss it.

- 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 Template feature lets you to use a simple x-script language to call properties and methods of the control at design as well at runtime. You can use this feature to build x-script strings to pass them at runtime.



For instance, the following template sample adds three properties and assign an icon to each of them, using the <img> tag.

```
BeginUpdate
Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl

Add("Icon1","",1).HTMLName = "1Icon <b>1</b>"
Add("Icon2","",1).HTMLName = "2Icon <b>2</b>"
Add("Icon3","",1).HTMLName = "3Icon <b>3</b>"
EndUpdate
```

property PropertiesList.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.

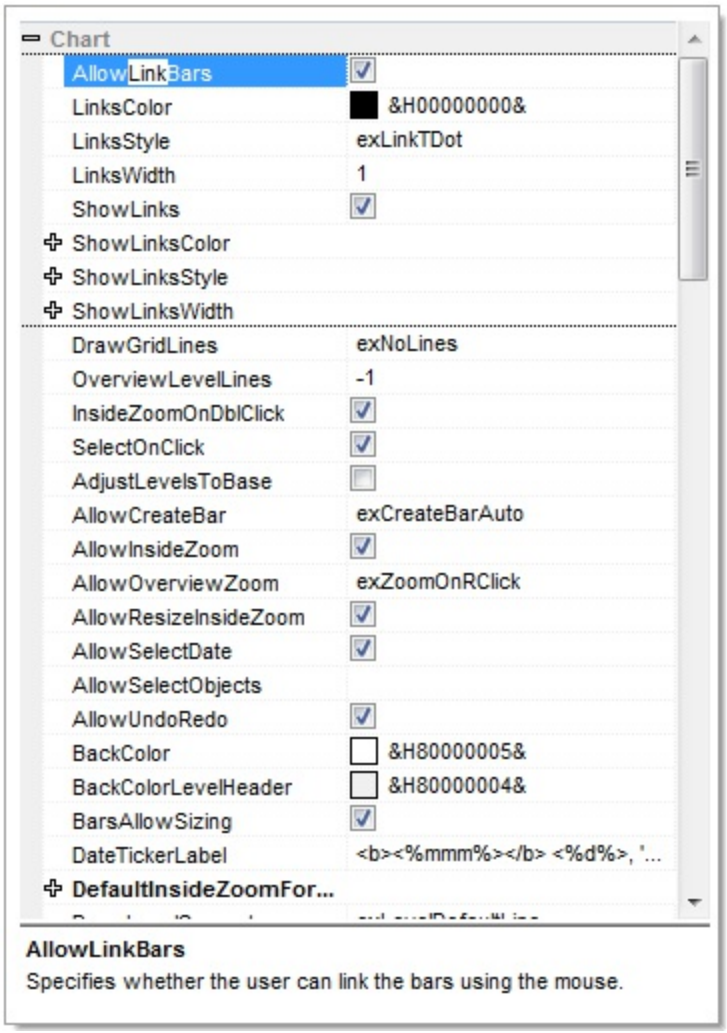
By default, the ImageSize property is 16 (pixels). The ImageSize property specifies the size of icons being loaded using the Images method. The control's Images collection is cleared if the ImageSize property is changed, so it is recommended to set the ImageSize property before calling the Images method. 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.

property PropertiesList.IncrementalSearch as AutoSearchEnum

Specifies whether the incremental search feature looks for starting of the property or if it contains the typed characters.

Type	Description
AutoSearchEnum	An AutoSearchEnum expression that specifies whether the control searches for the start of or if contains the typed characters.

By default, the IncrementalSearch property is exStartWith. Use the IncrementalSearch property to define a 'contains' incremental search. For instance, if the IncrementalSearch property is exContains + exMoveOnTop, the items are re-arranged so, the first items contain the typed characters, while the rest stay unchanged. In case the exMoveOnTop flag is included, the FilterBarPromptVisible property should be True, else it has no effect. Use the ExpandOnSearch property to automatically expand parent items as user types characters.



property PropertiesList.Indent as Long

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

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

Use the Indent property to increase or decrease the amount, in pixels, that child items are indented relative to their parent items. By default, the Indent property is 14 pixels. If the Indent property is 0 ,the control displays no indent for child items. Use the AutoIndent property to specify whether the control automatically indents the child items.

property PropertiesList.IndexItemsCollection as Long

Retrieves or sets a value that indicates the base index when control enumerates the items in the collection.

Type	Description
Long	A long expression that indicates the base index of the items in a collection.

By default, the IndexItemsCollection property is 0. Use the IndexItemsCollection to specify the base index of the items in a object of collection type. Use the NameItemsCollection property to specify the possible name of the properties that could indicate the caption of the items in the collection.

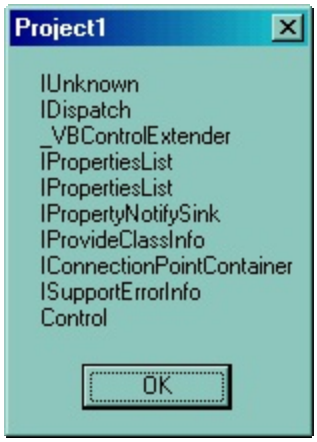
property PropertiesList.Interfaces (Object as IUnknown FAR*) as String

Retrieves the interfaces implemented by the object.

Type	Description
Object as IUnknown FAR*	An object being queried.
String	A string expression that indicates the list of the interfaces implemented by the object.

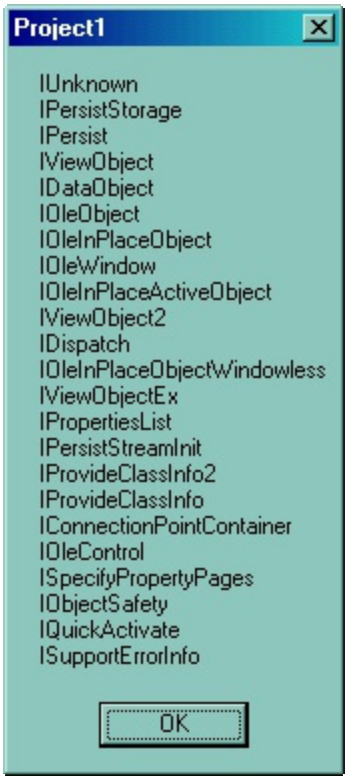
The following sample prints the interfaces implemented by the extended control object:

```
Private Sub Form_Load()
   MsgBox PropertiesList1.Interfaces(PropertiesList1)
End Sub
```



The following sample prints the interfaces implemented by the object itself:

```
Private Sub Form_Load()
   MsgBox PropertiesList1.Interfaces(PropertiesList1.Object)
End Sub
```


Project1
IUnknown
IPersistStorage
IPersist
IViewObject
IDataObject
IOleObject
IOleInPlaceObject
IOleWindow
IOleInPlaceActiveObject
IViewObject2
IDispatch
IOleInPlaceObjectWindowless
IViewObjectEx
IPropertiesList
IPersistStreamInit
IProvideClassInfo2
IProvideClassInfo
IConnectionPointContainer
IOleControl
ISpecifyPropertyPages
IObjectSafety
IQuickActivate
ISupportErrorInfo
OK

property PropertiesList.InvalidValueMessage as String

Retrieves or sets a value that indicates the error message displayed by browser when changing the property's value fails.

Type	Description
String	A string expression that indicates the error message that is displayed by the control when changing property's value fails.

No message occurs if the InvalidValueMessage is empty. By default the The InvalidValueMessage property is "Invalid property value."

property PropertiesList.Item (Index as Variant) as Property

Returns a Property object based on its index.

Type	Description
Index as Variant	A long expression that indicates the index of the Property being requested
Property	A Property object being accessed.

Use the Item property to access a property by its index. Use the Property property to access a Property giving its identifier. Use the Count property to get the number of items in the control. Use the Item and Count properties to enumerate the properties/items in the control.

The following sample enumerates the properties in the control:

```
With PropertiesList1
   Dim i As Long
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Name
   Next
End With
```

The following sample enumerates the properties in the control using the for each statement:

```
Dim p As EXPROPERTIESLISTLibCtl.Property
For Each p In PropertiesList1
   Debug.Print p.Name
Next
```

property PropertiesList.Layout as String

Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.

Type	Description
String	A String expression that specifies the control's layout.

You can use the Layout property to store the control's layout and to restore the layout later. For instance, you can save the control's Layout property to a file when the application is closing, and you can restore the control's layout when the application is loaded. The Layout property saves almost all of the control's properties that user can change at runtime (like changing the column's position by drag and drop). The Layout property does NOT save the control's data, so the Layout property should be called once you loaded the data from your database, xml or any other alternative. Once the data is loaded, you can call the Layout property to restore the View as it was saved. Before closing the application, you can call the Layout property and save the content to a file for reading next time the application is opened.

The Layout property saves/loads the following information:

- columns size and position
- current selection
- scrolling position and size
- expanded/collapsed properties, if any
- sorting columns
- filtering options

These properties are serialized to a string and encoded in BASE64 format.

The following movies show how Layout works:

- The Layout property is used to save and restore the control's view.

Generally, the Layout property can be used to save / load the control's layout (or as it is displayed). Thought, you can benefit of this property to sort the control using one or more columns as follows:

- multiplesort="";singlesort="", removes any previously sorting
- multiplesort="C3:1", sorts ascending the column with the index 3 (and add it to the sort bar if visible)
- singlesort="C4:2", sorts descending the column with the index 4 (it is not added to sort bar panel)
- multiplesort="C3:1";singlesort="C4:2", sorts ascending the column with the index 3 (

and add it to the sort bar if visible), and sorts descending the column with the index 4. In other words, it re-sort the control by columns 3 and 4.

- multiplesort="C3:1 C5:2";singlesort="C4:2", sorts ascending the column with the index 3 (and add it to the sort bar if visible), sorts descending the column with the index 5 (and add it to the sort bar if visible), and sorts descending the column with the index 4. In other words, it re-sort the control by columns 3, 5 and 4.

The format of the Layout in non-encoded form is like follows:

```
c0.filtertype=0
c0.position=0
c0.select=0
c0.visible=1
c0.width=96
....
columns=13
collapse="0-3 5-63 80-81 83"
filterprompt=""
focus=8
focuscolumnindex=0
hasfilter=1
hscroll=0
multiplesort="C12:1 C2:2"
searchcolumnindex=3
select="39 2 13 8"
selectcolumnindex=0
singlesort="C5:2"
treecolumnindex=0
vscroll=12
vscrolloffset=0
```

property PropertiesList.LinkCategories as Boolean

Retrieves or sets a value that indicates whether the categories are linked.

Type	Description
Boolean	A boolean expression that indicates whether the categories are linked.

By default the LinkCategories property is True. The LinkCategories has effect only if the ShowCategories is True. Use the MarkCategories property to mark categories.

property PropertiesList.MarkCategories as Boolean

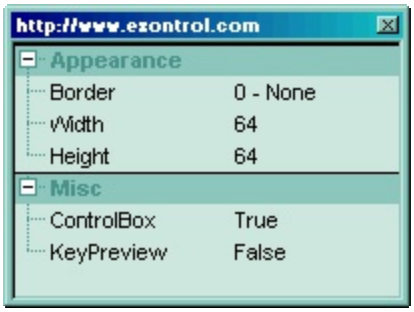
Specifies whether the categories are splited by separator lines.

Type	Description
Boolean	A boolean expression that indicates whether the categories are marked.

By default, the MarkCategories property is False.

The following sample shows how to split the categories using the MarkLineColor and MarkCategories properties:

```
With PropertiesList1
   .BeginUpdate
      .HasGridLines = False
      .DescriptionVisible = False
      .ShowCategories = True
      .LinkCategories = False
      .MarkCategories = True
      .MarkLineColor = vbBlack
      ' Adds a category
      .Add "Appearance", "", ReadOnly
      ' Adds items to 'Appearance' category
      With .Add("Border", 0, EditEnum, , "Appearance")
         .AddValue 0, "0 - None"
         .AddValue 1, "1 - Fixed"
      End With
      .Add "Width", 64, Edit, , "Appearance"
      .Add "Height", 64, Edit, , "Appearance"
      .ExpandItem("Appearance") = True
      ' Adds a category
      .Add "Misc", "", ReadOnly
      ' Adds items to 'Misc' category
      .Add "ControlBox", True, EditBoolean, , "Misc"
      .Add "KeyPreview", False, EditBoolean, , "Misc"
      .Refresh
   .EndUpdate
End With
```


http://www.ezontrol.com
Appearance
Border
0 - None
Width
64
Height
64
Misc
ControlBox
True
KeyPreview
False

property PropertiesList.MarkLineColor as Color

Retrieves or sets a value that indicates the color of lines that splits the categories.

Type	Description
Color	A color expression that indicates the color of lines that splits the categories.

Use the MarkLineColor property to specify the color of lines that splits the categories. Use the MarkLineColor property has effect only if the MarkCategories property is True and ShowCategories is True.

The following sample shows how to split the categories using the MarkLineColor and MarkCategories properties:

```
With PropertiesList1
   .BeginUpdate
      .HasGridLines = False
      .DescriptionVisible = False
      .ShowCategories = True
      .LinkCategories = False
      .MarkCategories = True
      .MarkLineColor = vbBlack
      ' Adds a category
      .Add "Appearance", "", ReadOnly
      ' Adds items to 'Appearance' category
      With .Add("Border", 0, EditEnum, , "Appearance")
         .AddValue 0, "0 - None"
         .AddValue 1, "1 - Fixed"
      End With
      .Add "Width", 64, Edit, , "Appearance"
      .Add "Height", 64, Edit, , "Appearance"
      .ExpandItem("Appearance") = True
      ' Adds a category
      .Add "Misc", "", ReadOnly
      ' Adds items to 'Misc' category
      .Add "ControlBox", True, EditBoolean, , "Misc"
      .Add "KeyPreview", False, EditBoolean, , "Misc"
      .Refresh
```

```
    .EndUpdate
End With
```

property PropertiesList.NameItemsCollection as String

Retrieves or sets a list of property's names separated by semicolon (;), that are used by properties browser when it requires a name for an item into a collection.

Type	Description
String	A string expression that indicates a list of property's names separated by semicolon (;), that are used by properties browser when it requires a name for an item into a collection.

By default, the NameItemsCollection is "Name;Caption;Item". Change the NameItemsCollection is your collection has different item's names. During loading the control uses the NameItemsCollection property to determine the name of each element into collection, if it is a collection of objects. If the ItemCollection property is True use the Object property to find the owner collection. For instance, if the collection contains only strings, the items added to browser's list will be numerated. Instead if the collection contains another objects, it uses the NameItemsCollection property to determine the caption that will be displayed on the name column. Use the IndexItemsCollection to specify the base index of the items in a object of collection type.

property PropertiesList.Option(Name as OptionEnum) as Variant

Specifies an option for the editor.

Type	Description
Name as OptionEnum	An OptionEnum expression that indicates the option being changed.
Variant	A Variant value that indicates the option's newly value.

Use the Option property to change particular options for a specified type editor. Use the Option property to customize the strings or behavior for different editors. For instance, you can specify the filter for EditFile properties, or specify the months for a drop down calendar control. The Option property applies the options to all editors, while the Option property of Property object may specify different options for different entries in the control. For instance, you can display a filter for some EditFile entries, and other filters for other EditFile entries in the same control.

In conclusion, you can specify options for the editors as follows:

- the same settings for all editors using Option property (by default).
- custom settings for the editor of an entry/property using the Property.Object property

The following VB sample customizes the EditDate editor to display strings in Romanian language:

```
With PropertiesList1
   .Option(exDateTodayCaption) = "Azi"
   .Option(exDateMonths) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August
Septembrie Octombrie Decembrie"
   .Option(exDateWeekDays) = "D L M M J V S"
   .Option(exDateFirstWeekDay) = 1
   .Add "Date", Date, EditDate
End With
```

The following VB sample changes the default filter for EditFile editors :

```
With PropertiesList1
   .Option(exEditFileFilter) = "INI Files|*.ini;*.init|All (*.*)|*.*"
   .Option(exEditFileTitle) = "Select an INI file"
   .Add "INI", "c:\temp\test.ini", EditFile, "Selects a file", "Custom"
End With
```

property PropertiesList.Property (Property as Variant) as Property

Gets a Property object given property's name or property's identifier.

Type	Description
Property as Variant	A string expression that indicates the property's name. or a long expression that indicates the property's identifier.
Property	A Property object being accessed.

Use the Property property to access at runtime to the control items. Use the Item property to access a Property by its index. Use the ID property to retrieve the property's identifier. The property's identifier is defined when adding new properties using the Add method. Use the SelectedProperty to retrieve the selected property.

property PropertiesList.ReadOnly as Boolean

Gets or sets whether the properties browser is read-only.

Type	Description
Boolean	A boolean expression that indicates whether the properties browser is read-only.

Use the ReadOnly property to disable editing properties. By default, the ReadOnly property is False. Use the Enabled property to disable the control. Use the Locked property to lock a property. Use the Enabled property to disable a property.

method PropertiesList.Refresh ()

Refreshes the properties values.

Type	Description

Refreshes the control. Use the BeginUpdate and EndUpdate methods to maintain performances while adding new entries. If a property is added manually, using the Add method, you need to call the Value property each time when you need to refresh the property's value. In case the properties list browses a COM object, using the Select method, you need to call the Refresh method to refresh the values for the properties in the browser, in case some changes occurs to the browsed object or if you need to. Also, if a property contains another COM object (EditObject type), the Refresh method updates the values for all browsed properties.

Use the Refresh method when adding values to a drop down editor like in the following sample:

```
With PropertiesList1
   .BeginUpdate
      .Add "Appearance", "", ReadOnly
      With .Add("Border", 0, EditEnum, , "Appearance")
         .AddValue 0, "0 - None"
         .AddValue 1, "1 - Fixed"
      End With
      .Refresh
   .EndUpdate
End With
```

If the Refresh method is not called in the above sample, the value for the Border property will be empty, because the predefined list of values for Border property were added after adding the property Border.

method PropertiesList.Remove (Property as Variant)

Removes a property from the list.

Type	Description
Property as Variant	A string expression that indicates the property's name, or a long expression that indicates the property's identifier.

The Remove method removes a property. The Remove method removes recursively the items/properties. Use Clear method if you need to clear the entire collection.

method PropertiesList.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. By default, the Icon parameter is 0, if it is missing.
Index as Variant	A long expression that indicates the index where icon is inserted. By default, the Index parameter is -1, if it is missing.

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 an image list to the control. The ImageSize property defines the size (width/height) of the icons within the control's Images collection.

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

i = PropertiesList1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), in this case the i specifies the index where the icon was added

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

i = PropertiesList1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), in this case the i is zero, because the first icon was replaced.

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

PropertiesList1.ReplaceIcon 0, i, in this case the i must be the index of the icon that follows to be removed

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

PropertiesList1.ReplaceIcon 0, -1

property PropertiesList.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 PropertiesList.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 PropertiesList.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 PropertiesList.ScrollHeight as Long

Specifies the height of the horizontal scrollbar.

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

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

property PropertiesList.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.

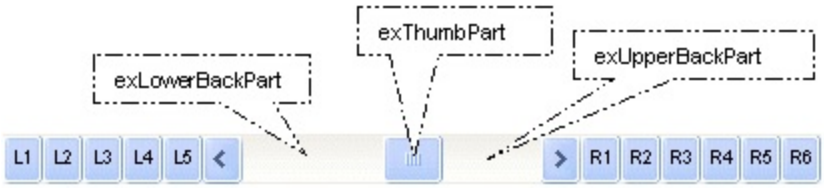
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

property PropertiesList.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.



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 PropertiesList1
  .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 AxPropertiesList1
   .BeginUpdate()
   .set_ScrollPartVisible(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part Or
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, True)
   .set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
   .set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.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 :

```
axPropertiesList1.BeginUpdate();
axPropertiesList1.set_ScrollPartVisible(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part |
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, true);
axPropertiesList1.set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axPropertiesList1.set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axPropertiesList1.EndUpdate();
```

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

```
m_propertiesList.BeginUpdate();
m_propertiesList.SetScrollBars( 15 /*exDisableBoth*/ );
m_propertiesList.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
```

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

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

```
With thisform.PropertiesList1
   .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 PropertiesList.ScrollPartCaptionAlignment(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as AlignmentEnum

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

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displayed
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the caption in the part of the scrollbar.

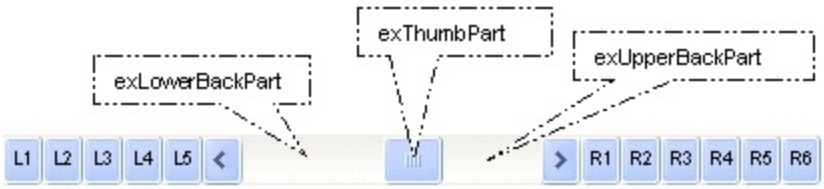
The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar. By default, the caption is centered. Use the ScrolPartCaption property to specify the caption being displayed on specified part of the scroll bar. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar.

property PropertiesList.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.

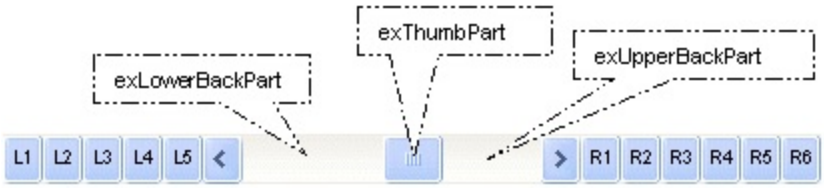


property PropertiesList.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

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.



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 PropertiesList1
  .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 AxPropertiesList1
   .BeginUpdate()
   .set_ScrollPartVisible(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part Or
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, True)
   .set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
   .set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.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 :

```
axPropertiesList1.BeginUpdate();
axPropertiesList1.set_ScrollPartVisible(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part |
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, true);
axPropertiesList1.set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axPropertiesList1.set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axPropertiesList1.EndUpdate();
```

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

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

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

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

```
With thisform.PropertiesList1
   .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 PropertiesList.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 PropertiesList.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.

property PropertiesList.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 PropertiesList.SelBackColor as Color

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

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

Use the SelBackColor and SelForeColor properties to define the colors for selected property. Use the HotForeColor property to specify a different foreground color for property when cursor hovers it. Use the HotBackColor property to specify a different background color for property when cursor hovers it.

method PropertiesList.Select (Object as Object)

Browses a new object to control.

Type	Description
Object as Object	An Object being browsed.

Use the Select method to browse the properties of a COM object. Use the SelectedObject property to browse properties of .NET objects (objects in the .NET framework). If the FireIncludeProperty is True, the Select method invokes the IncludeProperty event to let user filters the properties being browsed. Use the Add method to insert custom entries to the list.

The following **VB** sample browses the properties of the object itself (including the properties of the extended control, Visible, Top, and so on):

```
PropertiesList1.Select PropertiesList1
```

The following VB sample browses only the properties of the object itself:

```
PropertiesList1.Select PropertiesList1.Object
```

The following **C++** sample browses the control's properties:

```
IDispatch* pObject = NULL;
if ( SUCCEEDED( m_propertieslist.GetControlUnknown()->QueryInterface( IID_IDispatch,
(LPVOID*)&pObject ) ) )
{
   m_propertieslist.Select( pObject );
   pObject->Release();
}
```

In case you are using **ATL** (atlbase.h) classes you can use a code like follows:

```
CComQIPtr<IDispatch> spObject( m_propertieslist.GetControlUnknown() );
m_propertieslist.Select( spObject );
```

The following **VB.NET** sample browses the control's properties:

```
AxPropertiesList1.CtlSelect(AxPropertiesList1.GetOcx())
```

The following **C#** sample browses the control's properties:

```
axPropertiesList1.CtlSelect( axPropertiesList1.GetOcx() );
```

The following **VFP** sample browses the control's properties:

```
with thisform.PropertiesList1
   .Select(.Object)
endwith
```

If the Select method is called, and you need immediately after the list of browsed properties the following trick is required:

```
Private Type POINTAPI
        x As Long
        y As Long
End Type

Private Type MSG
    hwnd As Long
    message As Long
    wParam As Long
    lParam As Long
    time As Long
    pt As POINTAPI
End Type
Private Declare Function PeekMessage Lib "user32" Alias "PeekMessageA" (lpMsg As MSG,
ByVal hwnd As Long, ByVal wMsgFilterMin As Long, ByVal wMsgFilterMax As Long, ByVal
wRemoveMsg As Long) As Long
Private Const PM_NOREMOVE = &H0
Private Declare Function TranslateMessage Lib "user32" (lpMsg As MSG) As Long
Private Declare Function DispatchMessage Lib "user32" Alias "DispatchMessageA" (lpMsg
As MSG) As Long

' The list of properties is not immediately available, so we need to proceeds few messages
Private Sub waitSelect(ByVal h As Long)
    Dim m As MSG
    While PeekMessage(m, h, 0, 0, 1)
```

```
    TranslateMessage m
    DispatchMessage m
  Wend
End Sub
```

The following sample uses the trick, to expand the "Appearance" item:

```
Private Sub Form_Load()
  With PropertiesList1
    .BeginUpdate
      .HasLines = False
      .ShowCategories = True
      .MarkCategories = True
      .Select PropertiesList1.Object

      waitSelect .hwnd

      .ExpandItem("Appearance") = False
    .EndUpdate
  End With
End Sub
```

Note that if the waitSelect method is not called, the "Appearance" item is still expanded.

In VC++ the waitSelect method looks like follows:

```
// Function name  : waitSelect
// Description    : The list of properties is not immediately available, so we need to
proceeds few messages
// Return type    : void
// Argument       : HWND h
void waitSelect( HWND h )
{
  MSG m = {0};
  while ( PeekMessage( &m, h, 0, 0, PM_REMOVE ) )
  {
    TranslateMessage( &m );
    DispatchMessage( &m );
```

```
    }
}
```

The following VB sample browses the Form contains that hosts the ExPropertiesList control:

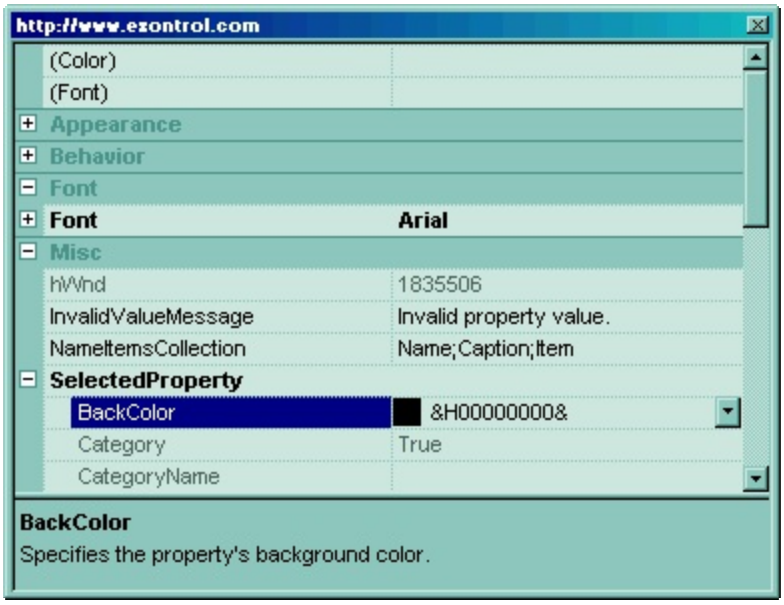
```
PropertiesList1.Select Me
```

The following VB sample clears the browsed object:

```
PropertiesList1.Select Nothing
```

The following VB sample browses an object and its categories:

```
Private Sub Form_Load()
    With PropertiesList1
        .BeginUpdate
            .HasLines = False
            .ShowCategories = True
            .MarkCategories = True
            .Select PropertiesList1.Object
        .EndUpdate
    End With
End Sub
```



property PropertiesList.SelectedObject as Variant

Browses a new object (com or .net) in the control.

Type	Description
Variant	A Variant expression that holds a COM object or a .NET object

Use the SelectedObject property to browse the properties of a .NET object. If the FireIncludeProperty is True, the Select method invokes the IncludeProperty event to let user filters the properties being browsed. Use the Add method to insert custom entries to the list. Use the Select method to browse properties of .COM objects.

The following VB.NET sample browses the properties of the form that contains the control:

```
AxPropertiesList1.SelectedObject = Me
```

The following C# sample browses the properties of the form that contains the control:

```
axPropertiesList1.SelectedObject = this;
```

property PropertiesList.SelectedProperty as Property

Retrieves a Property object that is currently selected.

Type	Description
Property	A Property object that is currently selected.

Use the SelChange event to notify your application when the current selection is changed. Use the Property property to retrieve a property giving its index or its name. Use the Selectable property to prevent a property to be selected.

The following sample prints the name and the type of the selected property (for instance, the sample is useful to find out the type of the property selected, when you need to include or exclude properties using the IncludeProperty event):

```
Private Sub PropertiesList1_SelChange()
    Debug.Print "You have selected the """ & PropertiesList1.SelectedProperty.Name & """.
The type for it is: " & PropertiesList1.SelectedProperty.Type
End Sub
```

property PropertiesList.SelForeColor as Color

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

Type	Description
Color	A color expression that indicates the selection foreground color.

Use the SelBackColor and SelForeColor properties to define the colors for selected property. Use the HotForeColor property to specify a different foreground color for property when cursor hovers it. Use the HotBackColor property to specify a different background color for property when cursor hovers it.

property PropertiesList.ShowCategories as Boolean

Retrieves or sets a value whether the browser includes the object categories.

Type	Description
Boolean	A boolean expression indicating whether the browser includes the object categories.

The ExPropertiesList control has the ability to categorize the object properties. Use the ShowCategories to display the object categories. Use the BackColorCategories and ForeColorCategories properties to define the background/foreground colors for category items. The LinkCategories specifies whether the control links the categories. Use the CategoryName property to get the property's category name. The Category property checks whether an items is category item or a property item. Use the DefaultCategory property to specify the default category. The default category includes all properties that have no category associated.

The following sample displays the control's categories:

```
Private Sub Form_Load()
   With PropertiesList1
      .HasLines = False
      .BackColorCategories = vbBlue
      .ForeColorCategories = vbWhite
      .ShowCategories = True
      .ShowPropertyPages = False
      .Select PropertiesList1.Object
   End With
End Sub
```

property PropertiesList.ShowHidden as Boolean

Retrieves or sets a value that indicates whether the properties browser displays the hidden members.

Type	Description
Boolean	A boolean expression that indicates whether the properties browser displays the hidden members.

Use the ShowHidden property to show hidden members. Changing the ShowHidden property at runtime invokes refreshing the control.

Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

property PropertiesList.ShowItemsCollection as Boolean

Retrieves or sets a value that indicates whether the properties browser includes the elements of a property that contains a collection.

Type	Description
Boolean	A boolean expression that indicates whether the properties browser includes the elements of a property that contains a collection.

Use the ShowItemsCollection property to expand properties that export collections. If you want to browse the collections, make sure that ShowObjects property is True. Changing the ShowItemsCollection property at runtime invokes refreshing the control.

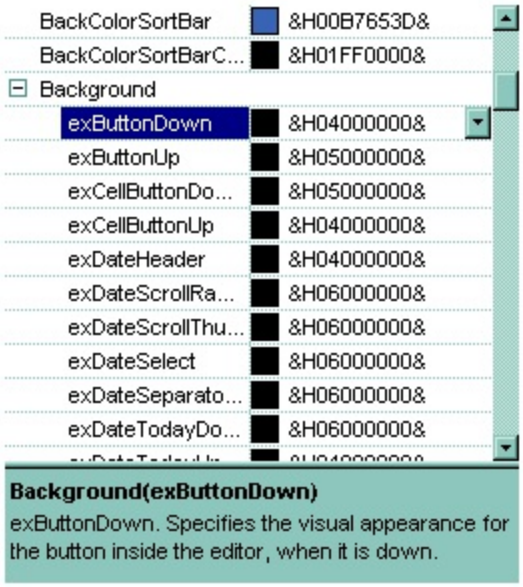
property PropertiesList.ShowMultipleParams as Boolean

Specifies whether the control loads properties with multiple parameters.

Type	Description
Boolean	A Boolean expression that indicates whether the control loads properties that have multiple parameters.

By default, the ShowMultipleParams property is True. Use the ShowMultipleParams property to include properties with multiple parameters when browsing a COM object, using the Select method. If the ShowMultipleParams property is True, only the properties with parameters of predefined type like enumeration or boolean are included. For instance, the property Grid.ItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, ColIndex as Long, HitTestInfo as HitTestInfoEnum) as HITEM can't be included in the browser, because it contains parameters that are not predefined like long type. The Name property indicates the name of the property. If the property includes multiple parameters, the Name property displays the name of the property and the parameters separated by comma, like "Background(exButtonDown)". If the property has a single parameter, the Description property indicates the description of the value in the predefined type. The Value property indicates the value of the property. Use the ToString property to save the control's properties and values to a string. Use the EditProperty option when adding a new property, to add properties one at the time, with none, one or more parameters.

For instance, let's say that we browse the eXGrid control that includes the "property **Grid.Background(Part as BackgroundPartEnum) as Color**. The property returns or sets a value that indicates the background color for parts in the control.". The Background property has a single parameter of BackgroundPartEnum type (enumeration type). The Background property retrieves different values based on the Part parameter. The Part parameter is of BackgroundPartEnum type. When the control includes the Background property, it combines all parameters with their values, and add a new property for each, as seen bellow.



This feature is new, and it is not available for other propertieslist controls. For instance, browses the ExPropertiesList with another browser, and check if the Option property is included in the browser? Definitely, this property may be browsed using the ExPropertiesList control.

property PropertiesList.ShowNonBrowsable as Boolean

Retrieves or sets a value that indicates whether the control displays the non browseable members.

Type	Description
Boolean	A boolean expression indicating whether the control displays the non browseable members.

Use the ShowNonBrowsable property to include non browseable members. Changing the ShowNonBrowsable property at runtime invokes refreshing the control.

Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

property PropertiesList.ShowObjects as Boolean

Retrieves or sets a value that indicates whether the properties browser includes the properties of object type.

Type	Description
Boolean	A boolean expression that indicates whether the properties browser includes the properties of object type.

Use the ShowObjects property to let the control browsing the properties of object type. Changing the ShowObjects at runtime invokes refreshing the control.

property PropertiesList.ShowPropertyPages as Boolean

Retrieves or sets a value that indicates whether the properties browser displays the object property pages.

Type	Description
Boolean	A boolean expression that indicates whether the properties browser displays the object property pages.

Use the ShowPropertyPages property to include the properties pages of the browsed object. Changing the ShowPropertyPages at runtime invokes refreshing the control.

Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

property PropertiesList.ShowReadOnly as Boolean

Retrieves or sets a value that indicates whether the properties browser displays the read only properties.

Type	Description
Boolean	A boolean expression that indicates whether the properties browser displays the read only properties.

Use the ShowReadOnly property to exclude properties that read only. Use the ReadOnly property to make the control editable. A read only member appears as grayed. Changing the ShowPropertyPages at runtime invokes refreshing the control. Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

property PropertiesList.ShowRestricted as Boolean

Retrieves or sets a value that indicates whether the properties browse displays the restricted members.

Type	Description
Boolean	A boolean expression that indicates whether the properties browse displays the restricted members.

Use the ShowRestricted property to include restricted members. Changing the ShowPropertyPages at runtime invokes refreshing the control.

Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

method PropertiesList.ShowToolTip (ToolTip as String, [Title as Variant], [Alignment as Variant], [X as Variant], [Y as Variant])

Shows the specified tooltip at given position.

Type	Description
ToolTip as String	The ToolTip parameter can be any of the following: • NULL(BSTR) or "<null>"(string) to indicate that the tooltip for the object being hovered is not changed • A String expression that indicates the description of the tooltip, that supports built-in HTML format (adds, replaces or changes the object's tooltip)
Title as Variant	The Title parameter can be any of the following: • missing (VT_EMPTY, VT_ERROR type) or "<null>" (string) the title for the object being hovered is not changed. • A String expression that indicates the title of the tooltip (no built-in HTML format) (adds, replaces or changes the object's title)
Alignment as Variant	A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing (VT_EMPTY, VT_ERROR) the alignment of the tooltip for the object being hovered is not changed. The Alignment parameter can be one of the following: • 0 - exTopLeft • 1 - exTopRight • 2 - exBottomLeft • 3 - exBottomRight • 0x10 - exCenter • 0x11 - exCenterLeft • 0x12 - exCenterRight • 0x13 - exCenterTop • 0x14 - exCenterBottom By default, the tooltip is aligned relative to the top-left corner (0 - exTopLeft).

X as Variant	Specifies the horizontal position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current horizontal position of the cursor (current x-position) • a numeric expression that indicates the horizontal screen position to show the tooltip (fixed screen x-position) • a string expression that indicates the horizontal displacement relative to default position to show the tooltip (moved)
Y as Variant	Specifies the vertical position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current vertical position of the cursor (current y-position) • a numeric expression that indicates the vertical screen position to show the tooltip (fixed screen y-position) • a string expression that indicates the vertical displacement relative to default position to show the tooltip (displacement)

Use the ShowToolTip method to display a custom tooltip at specified position or to update the object's tooltip, title or position. You can call the ShowToolTip method during the MouseMove event. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

For instance:

- ShowToolTip(`<null>`,`<null>`,,`+8`,`+8`), shows the tooltip of the object moved relative

to its default position
- ShowToolTip(`<null>`,`new title`), adds, changes or replaces the title of the object's tooltip
- ShowToolTip(`new content`), adds, changes or replaces the object's tooltip
- ShowToolTip(`new content`,`new title`), shows the tooltip and title at current position
- ShowToolTip(`new content`,`new title`,,`+8`,`+8`), shows the tooltip and title moved relative to the current position
- ShowToolTip(`new content`,``,,128,128), displays the tooltip at a fixed position
- ShowToolTip(``,``), hides the tooltip

The ToolTip parameter supports the built-in HTML format like follows:

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

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

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

height of the font. For instance the "<**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

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

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

property PropertiesList.ShowVariables as Boolean

Retrieves or sets a value that indicates whether the control displays the object variables.

Type	Description
Boolean	A boolean expression that indicates whether the control displays the object variables.

For instance, the properties of IFontDisp (font) type has variables like: Name, Size, and so on. Changing the ShowVariables at runtime executes a refresh of the browsed control. If you want to filter the object properties that has some special flags you can use Flags property of Property object.

Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

method PropertiesList.Sort ([Ascending as Variant], [Reserved as Variant])

Sorts the control.

Type	Description
Ascending as Variant	A boolean expression that indicates the sort order. True means ascending, False means descending.
Reserved as Variant	Reserved.

The Sort method sorts the properties. Use the SortObjects property to specify if the object properties should be placed on top or bottom side of the control once the user sorts a column. Use the SortOnClick property to specify whether a column gets sorted once the user clicks the column's header. Use the Sortable property to specify an un-sortable property.

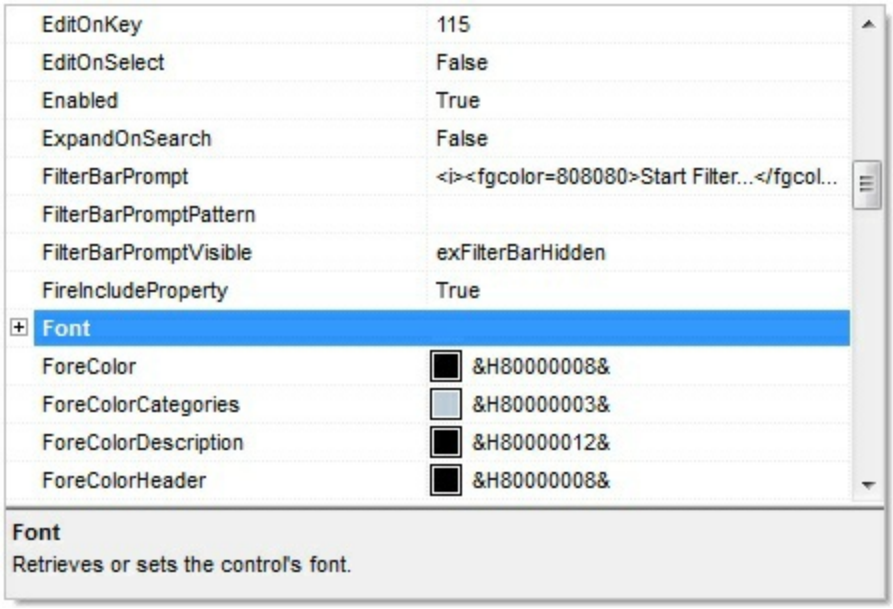
property PropertiesList.SortObjects as SortObjectsEnum

Specifies how the object properties are positioned once a Sort occurs.

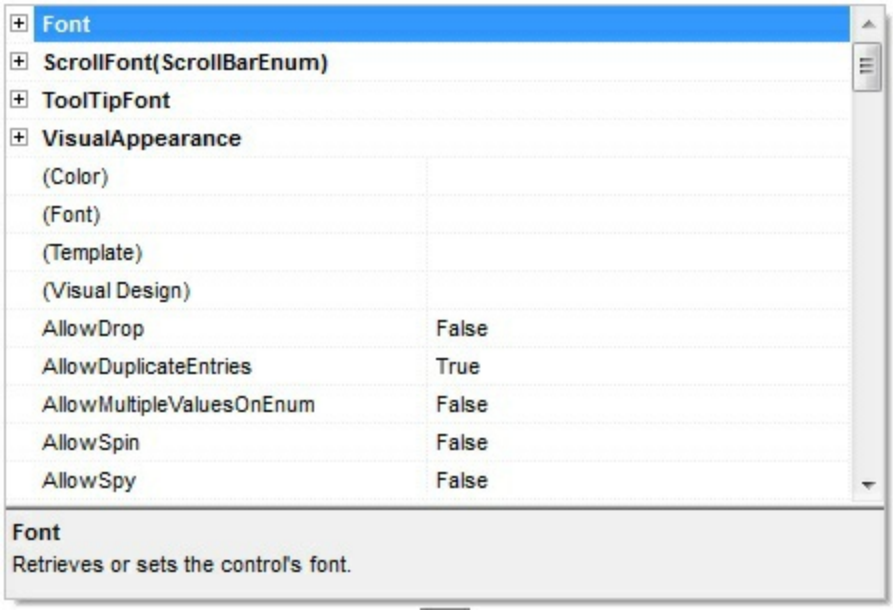
Type	Description
SortObjectsEnum	The SortObjectsEnum expression that specifies how the objects are positioned once the user sorts a column.

By default, the SortObjects property is exSortObjectsDefault. Use the SortObjects property to specify if the object properties should be placed on top or bottom side of the control once the user sorts a column. The Sort method sorts programmatically the control. Use the SortOnClick property to specify whether a column gets sorted once the user clicks the column's header. Use the Sortable property to specify an un-sortable property.

The following sample shows a sorted control, when SortObjects property is exSortObjectsDefault:



The following sample shows a sorted control, when SortObjects property is exSortObjectsTop:


Font
ScrollFont(ScrollBarEnum)
ToolTipFont
VisualAppearance
(Color)
(Font)
(Template)
(Visual Design)
AllowDrop
False
AllowDuplicateEntries
True
AllowMultipleValuesOnEnum
False
AllowSpin
False
AllowSpy
False
Font
Retrieves or sets the control's font.

property PropertiesList.SortOnClick as SortOnClickEnum

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

Type	Description
SortOnClickEnum	A SortOnClickEnum expression that indicates whether the control sorts the columns when clicking the control's header.

By default, the SortOnClick property is exDefaultSort, that means that the control sorts the column's being clicked. Use the SortOnClick property to disable sorting items when the user clicks on the column's header. Use the HeaderVisible property to show or hide the control's header. Use the ColumnsAllowSizing property to specify whether the user can resize a column at runtime. Use the Sortable property to specify an un-sortable property.

property PropertiesList.Template as String

Specifies the control's template.

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

The control's template uses the X-Script language to initialize the control's content. Use the Template property page of the control to update the control's Template property. Use the Template property to execute code by passing instructions as a string (template string). Use the ExecuteTemplate property to execute a template script and gets the result.

Most of our UI components provide a Template page that's accessible in design mode. No matter what programming language you are using, you can have a quick view of the component's features using the WYSWYG Template editor.

- Place the control to your form or dialog.
- Locate the Properties item, in the control's context menu, in design mode. If your environment doesn't provide a Properties item in the control's context menu, please try to locate in the Properties browser.
- Click it, and locate the Template page.
- Click the Help button. In the left side, you will see the component, in the right side, you will see a x-script code that calls methods and properties of the control.

The control's Template page helps user to initialize the control's look and feel in design mode, using the x-script language that's easy and powerful. The Template page displays the control on the left side of the page. On the right side of the Template page, a simple editor is displayed where user writes the initialization code. The control's look and feel is automatically updated as soon as the user types new instructions. The Template script is saved to the container persistence (when Apply button is pressed), and it is executed when the control is initialized at runtime. Any component that provides a WYSWYG Template page, provides a Template property. The Template property executes code from a string (template string).

The Template 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 PropertiesList.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 PropertiesList.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 PropertiesList.ToolTipDelay as Long

Specifies the time in ms that passes before the ToolTip appears.

Type	Description
Long	A long expression that specifies the time in ms that passes before the ToolTip appears.

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip.

property PropertiesList.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

Type	Description
IFontDisp	A Font object being used to display the tooltip.

Use the ToolTipFont property to assign a font for the control's tooltip. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window.

property PropertiesList.ToolTipPopDelay as Long

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

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

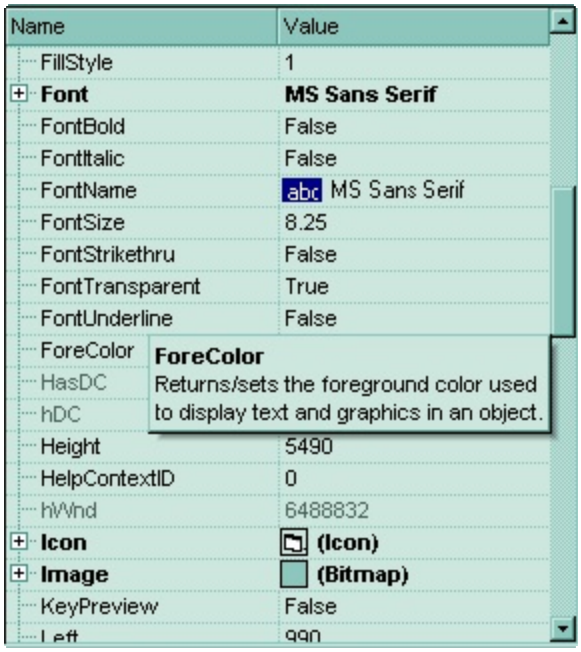
If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to assign a font for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

property PropertiesList.ToolTipWidth as Long

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

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

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



property PropertiesList.ToString (Type as ToStringEnum) as String

Saves the control's content to a string, as it is displayed.

Type	Description
Type as ToStringEnum	A ToStringEnum expression that indicates whether the values of the properties use literals or numbers.
String	A string expression that indicates the list of properties with their values.

The ToString property gets the list of properties with their values to a string, as they are displayed. Use the ToString property to save the properties and their values to a string for the object being browsed. The ToString(exLiterals) property lists the properties closely as being displayed in the control. The ToString(exNumbers) property lists the properties being displayed in the control, after the predefined literals are replaced. Use the Select method to browse a COM object. Use the ExpandAll method to expand all items in the control.

Here's how the properties are listed, when the control browses the eXGrid control:

```
(Color)
(Font)
(Template)
Appearance = Sunken
ASCIILower = "abcdefghijklmnopqrstuvwxyzüéâäŕĺçęëčdďîěôöňűůáíóúń"
ASCIIUpper = "ABCDEFGHIJKLMNOPQRSTUVWXYZÜÉÂÄŔĹÇĘËČĎÎĚÔÖŇŰŮÁÍÓÚŃ"
AutoEdit = True
AutoSearch = True
BackColor = &H80000005;&
BackColorAlternate = &H00000000;&
BackColorHeader = &H8000000F;&
BackColorLevelHeader = &H8000000F;&
BackColorLock = &H80000005;&
BackColorSortBar = &H80000010;&
BackColorSortBarCaption = &H8000000F;&
Background(exButtonDown) = &H00000000;&
Background(exButtonUp) = &H00000000;&
Background(exCellButtonDown) = &H00000000;&
Background(exCellButtonUp) = &H00000000;&
Background(exDateHeader) = &H00000000;&
Background(exDateScrollRange) = &H00000000;&
```

```
Background(exDateScrollThumb) = &H00000000;&
Background(exDateSelect) = &H00000000;&
Background(exDateSeparatorBar) = &H00000000;&
Background(exDateTodayDown) = &H00000000;&
Background(exDateTodayUp) = &H00000000;&
Background(exDropDownButtonDown) = &H00000000;&
Background(exDropDownButtonUp) = &H00000000;&
Background(exFooterFilterBarButton) = &H00000000;&
Background(exHeaderFilterBarButton) = &H00000000;&
Background(exSelBackColorFilter) = &H00000000;&
Background(exSelectInPlace) = &H00000000;&
Background(exSelForeColorFilter) = &H00000000;&
Background(exSliderRange) = &H00000000;&
Background(exSliderThumb) = &H00000000;&
Background(exSpinDownButtonDown) = &H00000000;&
Background(exSpinDownButtonUp) = &H00000000;&
Background(exSpinUpButtonDown) = &H00000000;&
Background(exSpinUpButtonUp) = &H00000000;&
CauseValidateValue = False
ColumnAutoResize = True
Columns
   Column 1
      Alignment = LeftAlignment
      AllowDragging = True
      AllowSizing = True
      AllowSort = True
      AutoSearch = exStartWith
      [AutoWidth = 24]
      Caption = "Column 1"
      Data
      Def(exCellBackColor)
      Def(exCellButtonAutoWidth) = 0
      Def(exCellForeColor)
      Def(exCellFormatLevel) = ""
      Def(exCellHasButton) = 0
      Def(exCellHasCheckBox) = 0
      Def(exCellHasRadioButton) = 0
```

```
Def(exCellSingleLine) = -1
Def(exCellValueFormat) = 0
DefaultSortOrder = False
DisplayFilterButton = False
DisplayFilterDate = False
DisplayFilterPattern = True
DisplaySortIcon = True
Editor
   Appearance = NoApp
   ButtonWidth = 13
   DropDownAlignment = LeftAlignment
   DropDownAutoWidth = exDropDownAutoWidth
   DropDownMinWidth = 164
   DropDownRows = 7
   DropDownVisible = True
   EditType = EditType
   Locked = False
   Mask = ""
   MaskChar = 95
   Numeric = exAllChars
   Option(exAutoDropDownList) = 0
   Option(exAutoSearch) = 0
   Option(exCalcButtonHeight) = 24
   Option(exCalcButtons) = "7,8,9,/,C\r\n4,5,6,*,1/x\r\n1,2,3,-,sqrt\r\n0,+/-,.,+,="
   Option(exCalcButtonWidth) = 24
   Option(exCalcCannotDivideByZero) = "Cannot divide by zero."
   Option(exCalcExecuteKeys) = -1
   Option(exCalcPictureDown) = ""
   Option(exCalcPictureUp) = ""
   Option(exCheckValue0) = 0
   Option(exCheckValue1) = 1
   Option(exCheckValue2) = 2
   Option(exColorListShowName) = 0
   Option(exColorShowPalette) = -1
   Option(exColorShowSystem) = -1
   Option(exDateAllowNullDate) = -1
   Option(exDateFirstWeekDay) = 0
```

```
        Option(exDateMarkToday) = 0
        Option(exDateMonths) = "January February March April May June July August
September October November December"
        Option(exDateShowScroll) = -1
        Option(exDateShowTodayButton) = -1
        Option(exDateTodayCaption) = "Today"
        Option(exDateWeekDays) = "S M T W T F S"
        Option(exDateWeeksHeader) = 0
        Option(exDownArrow) = -1
        Option(exDropDownImage) = -1
        Option(exEditDecimalSymbol) = 46
        Option(exEditLimitText) = 0
        Option(exEditLockedBackColor) = -2147483633
        Option(exEditLockedForeColor) = 0
        Option(exEditPassword) = 0
        Option(exEditPasswordChar) = 42
        Option(exEditRight) = 0
        Option(exEditSelLength) = -1
        Option(exEditSelStart) = 0
        Option(exEndKey) = -1
        Option(exExpandOnSearch) = 0
        Option(exHomeKey) = -1
        Option(exKeepSelBackColor) = 0
        Option(exLeftArrow) = -1
        Option(exMemoAutoSize) = -1
        Option(exMemoDropDownAcceptReturn) = -1
        Option(exMemoDropDownHeight) = 116
        Option(exMemoDropDownWidth) = 128
        Option(exMemoHScrollBar) = 0
        Option(exMemoVScrollBar) = 0
        Option(exPageDownKey) = -1
        Option(exPageUpKey) = -1
        Option(exProgressBarAlignment) = 0
        Option(exProgressBarBackColor) = -2147483635
        Option(exProgressBarMarkTicker) = -1
        Option(exRightArrow) = -1
        Option(exShowPictureType) = -1
```

```
    Option(exSliderMax) = 100
    Option(exSliderMin) = 0
    Option(exSliderStep) = 1
    Option(exSliderWidth) = 64
    Option(exSpinStep) = 1
    Option(exUpArrow) = -1
    PartialCheck = False
    PopupAppearance = ShadowApp
    UserEditorObject
Enabled = True
Filter = ""
FilterBarDropDownWidth = 1
FilterList = exAllItems
FilterType = exAll
FireFormatColumn = False
FormatLevel = ""
HeaderAlignment = LeftAlignment
HeaderBold = False
HeaderImage = 0
HeaderImageAlignment = LeftAlignment
HeaderItalic = False
HeaderStrikeOut = False
HeaderUnderline = False
HeaderVertical = False
HTMLCaption = ""
[Index = 0]
Key = ""
LevelKey
MaxWidthAutoResize = -1
MinWidthAutoResize = 0
PartialCheck = False
Position = 0
Selected = False
SortOrder = SortNone
SortPosition = -1
SortType = SortString
ToolTip = "..."
```

```
      Visible = True
      Width = 269
      WidthAutoResize = False
   [Count = 1]
ColumnsAllowSizing = False
ContinueColumnScroll = True
CountLockedColumns = 0
DataSource
DefaultEditorOption(exAutoDropDownList) = 0
DefaultEditorOption(exAutoSearch) = 0
DefaultEditorOption(exCalcButtonHeight) = 24
DefaultEditorOption(exCalcButtons) = "7,8,9,/,C\r\n4,5,6,*,1/x\r\n1,2,3,-,sqrt\r\n0,+/-
,.,+,="
DefaultEditorOption(exCalcButtonWidth) = 24
DefaultEditorOption(exCalcCannotDivideByZero) = "Cannot divide by zero."
DefaultEditorOption(exCalcExecuteKeys) = -1
DefaultEditorOption(exCalcPictureDown) = ""
DefaultEditorOption(exCalcPictureUp) = ""
DefaultEditorOption(exCheckValue0) = 0
DefaultEditorOption(exCheckValue1) = 1
DefaultEditorOption(exCheckValue2) = 2
DefaultEditorOption(exColorListShowName) = 0
DefaultEditorOption(exColorShowPalette) = -1
DefaultEditorOption(exColorShowSystem) = -1
DefaultEditorOption(exDateAllowNullDate) = -1
DefaultEditorOption(exDateFirstWeekDay) = 0
DefaultEditorOption(exDateMarkToday) = 0
DefaultEditorOption(exDateMonths) = "January February March April May June July
August September October November December"
DefaultEditorOption(exDateShowScroll) = -1
DefaultEditorOption(exDateShowTodayButton) = -1
DefaultEditorOption(exDateTodayCaption) = "Today"
DefaultEditorOption(exDateWeekDays) = "S M T W T F S"
DefaultEditorOption(exDateWeeksHeader) = 0
DefaultEditorOption(exDownArrow) = -1
DefaultEditorOption(exDropDownImage) = -1
DefaultEditorOption(exEditDecimalSymbol) = 46
```

DefaultEditorOption(exEditLimitText) = 0
DefaultEditorOption(exEditLockedBackColor) = -2147483633
DefaultEditorOption(exEditLockedForeColor) = 0
DefaultEditorOption(exEditPassword) = 0
DefaultEditorOption(exEditPasswordChar) = 42
DefaultEditorOption(exEditRight) = 0
DefaultEditorOption(exEditSelLength) = -1
DefaultEditorOption(exEditSelStart) = 0
DefaultEditorOption(exEndKey) = -1
DefaultEditorOption(exExpandOnSearch) = 0
DefaultEditorOption(exHomeKey) = -1
DefaultEditorOption(exKeepSelBackColor) = 0
DefaultEditorOption(exLeftArrow) = -1
DefaultEditorOption(exMemoAutoSize) = -1
DefaultEditorOption(exMemoDropDownAcceptReturn) = -1
DefaultEditorOption(exMemoDropDownHeight) = 116
DefaultEditorOption(exMemoDropDownWidth) = 128
DefaultEditorOption(exMemoHScrollBar) = 0
DefaultEditorOption(exMemoVScrollBar) = 0
DefaultEditorOption(exPageDownKey) = -1
DefaultEditorOption(exPageUpKey) = -1
DefaultEditorOption(exProgressBarAlignment) = 0
DefaultEditorOption(exProgressBarBackColor) = -2147483635
DefaultEditorOption(exProgressBarMarkTicker) = -1
DefaultEditorOption(exRightArrow) = -1
DefaultEditorOption(exShowPictureType) = -1
DefaultEditorOption(exSliderMax) = 100
DefaultEditorOption(exSliderMin) = 0
DefaultEditorOption(exSliderStep) = 1
DefaultEditorOption(exSliderWidth) = 64
DefaultEditorOption(exSpinStep) = 1
DefaultEditorOption(exUpArrow) = -1
DefaultItemHeight = 18
Description(exFilterBarAll) = "(All)"
Description(exFilterBarAnd) = " and "
Description(exFilterBarBlanks) = "(Blanks)"
Description(exFilterBarChecked) = "(Checked)"

Description(exFilterBarDate) = "Date:"
Description(exFilterBarDateMonths) = "January February March April May June July August September October November December"
Description(exFilterBarDateTitle) = "Date"
Description(exFilterBarDateTo) = "to"
Description(exFilterBarDateTodayCaption) = "Today"
Description(exFilterBarDateTooltip) = "You can filter the items into a given interval of dates. For instance, you can filter all items dated before a specified date (**to 2/13/2004**), or all items dated after a date (**Feb 13 2004 to**) or all items that are in a given interval (**2/13/2004 to 2/13/2005**)."
Description(exFilterBarDateWeekDays) = "S M T W T F S"
Description(exFilterBarFilterForCaption) = "Filter For:"
Description(exFilterBarFilterForTooltip) = "A pattern filter may contain the wild card characters '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character, '|' determines the options in the pattern. For instance: '1*|2*' specifies all items that start with '1' or '2'. If the filter is of numeric type you can filter numbers giving numeric rules. For instance, ">10 <100" filter indicates all numbers greater than 10 and less than 100."
Description(exFilterBarFilterTitle) = "Filter"
Description(exFilterBarIsBlank) = "IsBlank"
Description(exFilterBarIsChecked) = "IsChecked"
Description(exFilterBarIsNonBlank) = "not IsBlank"
Description(exFilterBarIsUnchecked) = "not IsChecked"
Description(exFilterBarNonBlanks) = "(NonBlanks)"
Description(exFilterBarPatternFilterTitle) = "Pattern/Numeric Filter"
Description(exFilterBarPatternTooltip) = "You can select multiple filter items as many as you like by keeping the CTRL key pressed. Start typing characters if you like to enter a filter as a pattern that may include wild card characters like *,? or #. Press ENTER key to filter the items using the typed pattern. If the filter is of numeric type you can filter numbers giving numeric rules. For instance, ">10 <100" filter indicates all numbers greater than 10 and less than 100."
Description(exFilterBarTooltip) = "You can select multiple filter items as many as you like by keeping the CTRL key pressed. "
Description(exFilterBarUnchecked) = "(Unchecked)"
DetectAddNew = False
DetectDelete = False
DrawGridLines = exNoLines

```
[Editing = 0]
Enabled = True
EnsureOnSort = True
ExpandOnDblClick = True
ExpandOnKeys = True
ExpandOnSearch = False
FilterBarBackColor = &H8000000F;&
FilterBarCaption = ""
FilterBarDropDownHeight = 0.5
FilterBarFont
   Bold = True
   Charset = 0
   Italic = False
   Name = "Arial"
   Size = 8.25
   Strikethrough = False
   Underline = False
   Weight = 700
FilterBarForeColor = &H80000008;&
FilterBarHeight = -1
FilterInclude = exItemsWithoutChilds
FocusColumnIndex = 0
Font
   Bold = False
   Charset = 0
   Italic = False
   Name = "Arial"
   Size = 8.25
   Strikethrough = False
   Underline = False
   Weight = 400
ForeColor = &H80000008;&
ForeColorHeader = &H80000008;&
ForeColorLock = &H80000008;&
ForeColorSortBar = &H80000010;&
FullRowSelect = exItemSel
GridLineColor = &H00889048;&
```

HasButtons = exPlus
HasButtonsCustom(False) = 0
HasButtonsCustom(True) = 0
HasLines = exDotLine
HeaderAppearance = Raised
HeaderHeight = 18
HeaderVisible = True
HideSelection = False
[hWnd = 4391834]
HyperLinkColor = &H00FF6531;&
Indent = 22
Items
  [(000) = 68282272]
  DefaultItem = 0
  [FirstVisibleItem = 68282272]
  [FocusItem = 68282272]
  [ItemCount = 1]
  LockedItemCount(exBottom) = 0
  LockedItemCount(exMiddle) = 0
  LockedItemCount(exTop) = 0
  PathSeparator = "\"
  [RootCount = 1]
  [SelectCount = 1]
  SelectPos = 0
  [VisibleCount = 1]
ItemsAllowSizing = False
LinesAtRoot = exNoLinesAtRoot
MarkSearchColumn = True
MarkTooltipCells = False
MarkTooltipCellsImage = 0
OLEDropMode = exOLEDropNone
Picture
PictureDisplay = Tile
PictureDisplayLevelHeader = Tile
PictureLevelHeader
RadioImage(False) = 0
RadioImage(True) = 0

RClickSelect = False
ReadOnly = exReadWrite
ScrollBars = exBoth
ScrollBySingleLine = False
ScrollPos(False) = 0
ScrollPos(True) = 0
SearchColumnIndex = 0
SelBackColor = &H8000000D;&
SelBackMode = exOpaque
SelectByDrag = True
SelectColumnIndex = 0
SelectColumnInner = 0
SelForeColor = &H8000000E;&
ShowFocusRect = True
ShowImageList = False
ShowLockedItems = True
SingleSel = True
SingleSort = True
SortBarCaption = "Drag a **column** header here to sort by that column."
SortBarColumnWidth = -96
SortBarHeight = 18
SortBarVisible = False
SortOnClick = exDefaultSort
Template = "BeginUpdate\r\nColumns\r\n{\r\n\t"Column
1"\r\n\t{\r\n\t\tEditor\r\n\t\t{\r\n\t\t\tEditType =
1\r\n\t\t}\r\n\t}\r\n}\r\nItems\r\n{\r\n\tDim h\r\n\th = AddItem(16)\r\n\tSelectItem(h) =
True\r\n}\r\nEndUpdate"
TooltipCellsColor = &H00FF6531;&
ToolTipDelay = 500
ToolTipPopDelay = 5000
ToolTipWidth = 196
TreeColumnIndex = 0
UnboundHandler
UseTabKey = True
Version = "3.1.0.3.DEBUG"
VirtualMode = False
VisualAppearance

property PropertiesList.UseVisualTheme as UIVisualThemeEnum

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

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

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

property PropertiesList.Version as String

Retrieves the control's version.

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

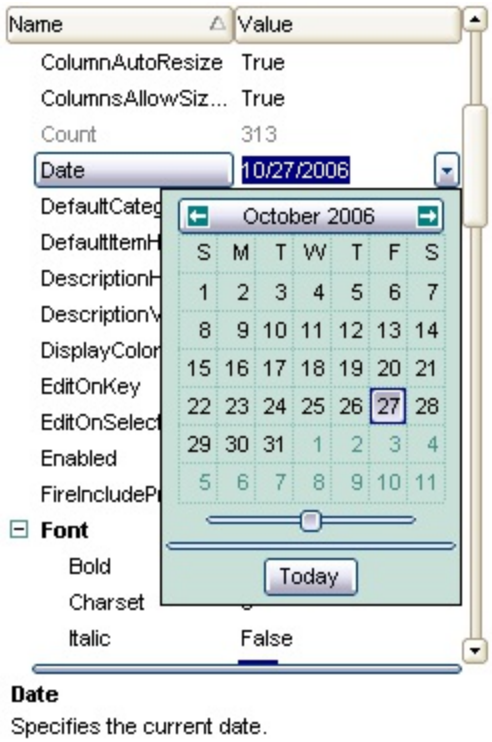
The Version property retrieves the control's version.

property PropertiesList.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.



property PropertiesList.VisualDesign as String

Invokes the control's VisualAppearance designer.

Type	Description
String	A String expression that encodes the control's Visual Appearance.

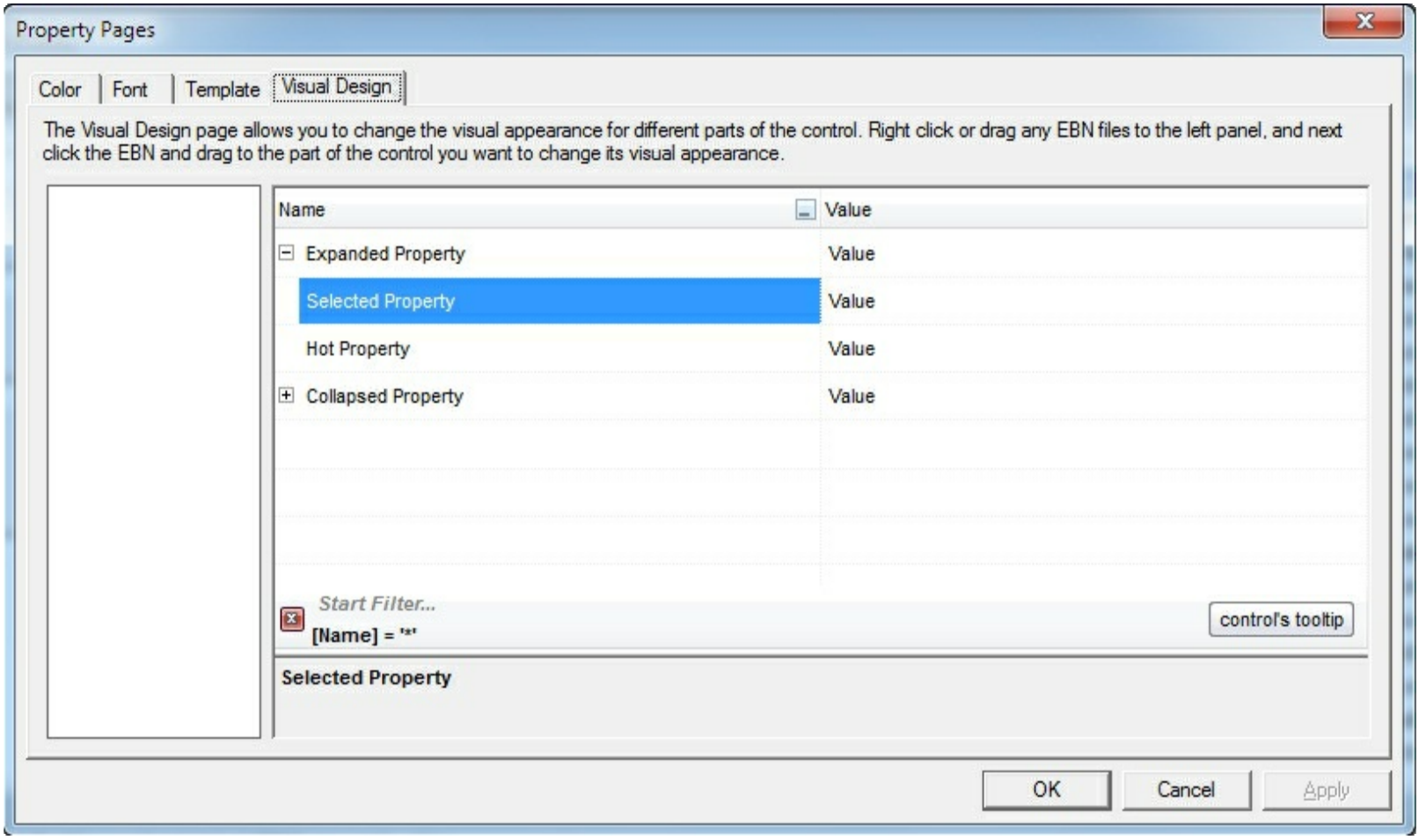
By default, the VisualDesign property is "". The VisualDesign property helps you to define fast and easy the control's visual appearance using the XP-Theme elements or EBN objects. The VisualDesign property can be accessed on design mode, and it can be used to design the visual appearance of different parts of the control by drag and drop XP or EBN elements. The VisualAppearance designer returns an encoded string that can be used to define different looks, just by calling the VisualDesign = encoded_string. If you require removing the current visual appearance, you can call the VisualDesign on "" (empty string). The VisualDesign property encodes EBN or XP-Theme nodes, using the Add method of the Appearance collection being accessed through the VisualAppearance property.

- For the /COM version, click the control in Design mode, select the Properties, and choose the "Visual Design" page.

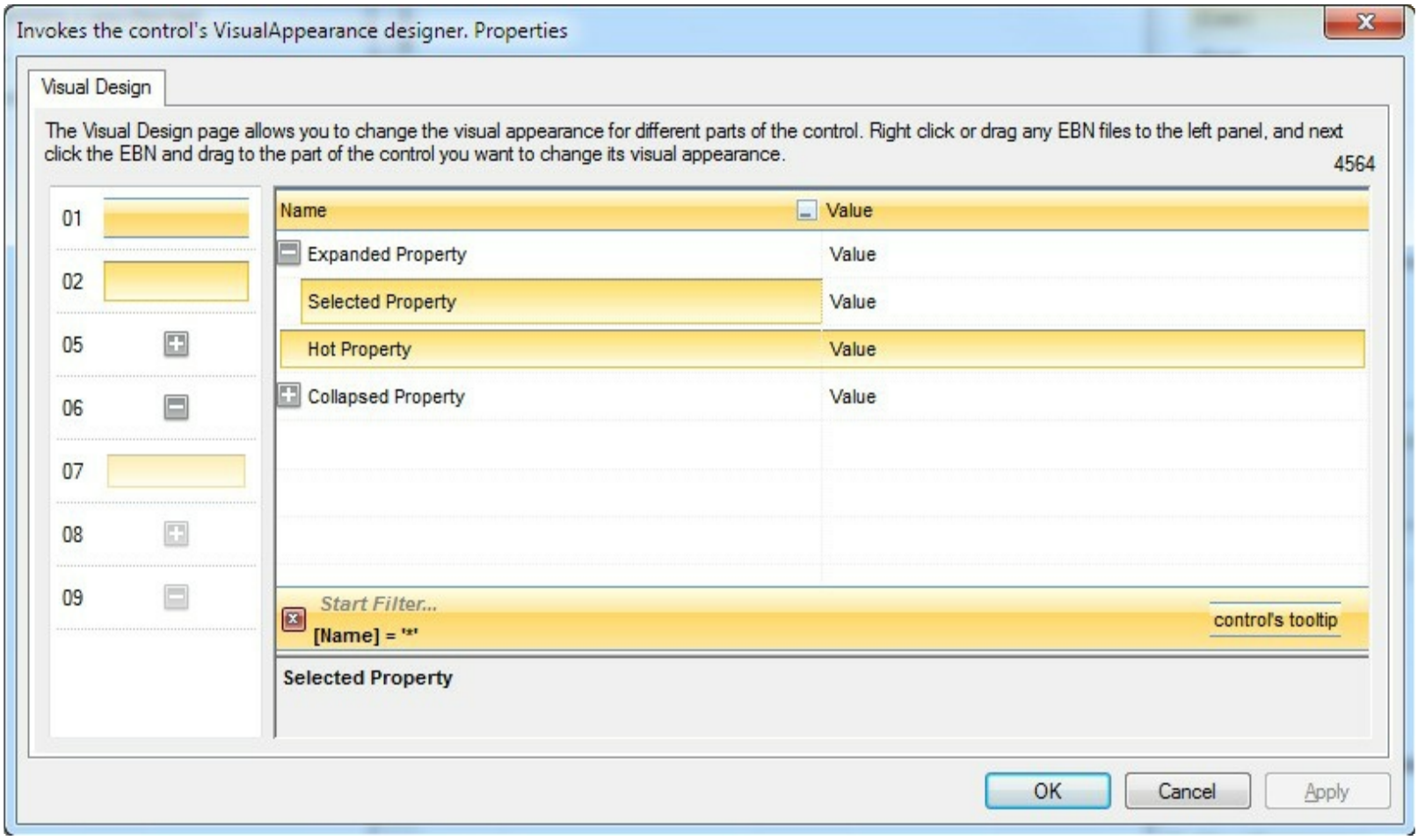
- Click here to watch a movie on how you define the control's visual appearance using the XP-Theme
- Click here to watch a movie on how you define the control's visual appearance using the EBN files.

The left panel, should be user to add your EBN or XP-Theme elements. Once you add them drag and drop the EBN or XP-Theme element from the left side to the part which visual appearance you want to change.

The following picture shows the control's VisualDesign form (empty):



The following picture shows the control's VisualDesign form after applying some EBN objects:



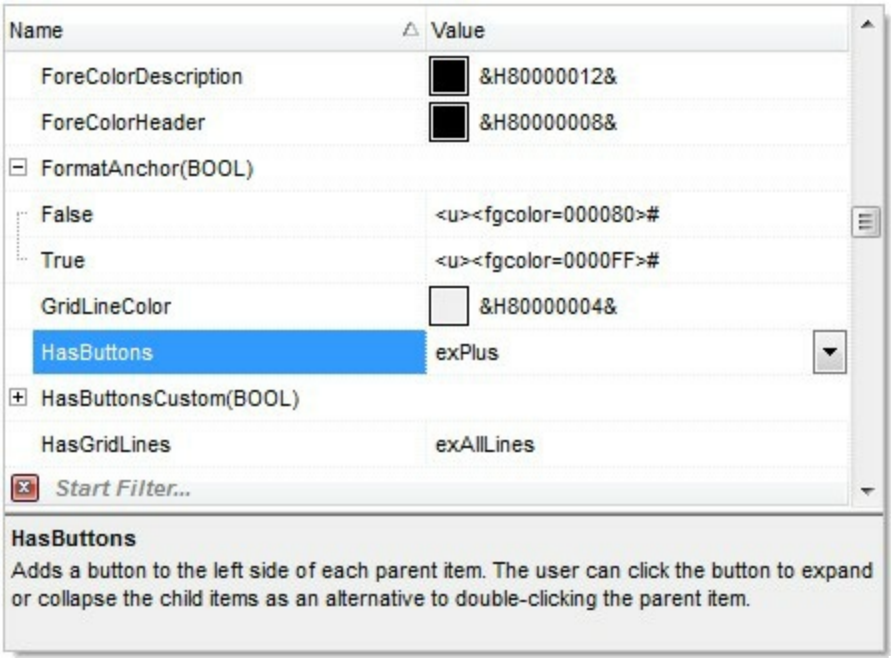
This layout generates the following code:

```
With PropertiesList1
   .VisualDesign =
"gBFLBWIgBAEHhEJAEGg6pBcHF4eAoABslfj/jEJAcKhYEjgCicKA4GjgBA8cAgIjgDB0cBAPjgJ
 & _
"MAAREEHf0bAECIROoEcn8IALlo05YAQGKIYBkAKBQAGaAoDDQOQ5QwAAwjCK8EwsACEI
 & _
"kAYQEiKLoaRzAcwyDAcQRFCKUJxlKa5PjaL40UBFRipGx4ASHIqLJygOSaLigAo/SAEUZ1FKs
 & _
"jBT8EzHMqeZ7qSgZfiKSgAUxcEpyJZsXxnGaQJ6jKaJcrKfZxW5GYgRWINLQvLCsKRvezxQo+
 & _
"AAKO4cjOWIshEHAMGsJAAjWZRujoS51DcFxQg2XoABobRUGsGYMkcS4qDgbIOllbwcjCb
 & _
"gSFANCYM4hlSZBlBiSANkAeAZAiG5ag0DYhCKCBOiISgYguDg7HyRh6EyS5sncex9n8L4PlO
 & _
"Gf4AGASAOAaAIgggFgJgGYIoAUAwmk4U5+BCBIhBg+4FGGCBSBqBZhjgZgcgSUAyGCZA
 & _
"aIVAMJRODOfg0g2IxYl4OoOGOSJiD6DpjnidhAg3TFeD+ZBpCIR4SGSSQuE6EoIAkNhRhOZ
 & _
"gJHIYoXmYOYGGSFtMDIWYag+eYyG6GpnAmNhxhuZwplYdIdmWFAMgkOF6G4fYfCeak8i
 & _
"6DqEIkmkKhWhGJZpEoYoWiYaQEAyAg3g+eh+4oKgqhaJIpmoOomiaKoqgqLoiECR5zHSf
 & _
"GsRAMHwNwPnsDpFjFpQmkiMpsHsNpKjOLJLD6RmihsEpVD+bR7GaVo2m2CxylqNxuAsfp
 & _
"wC42nGO5vCvbIck2I5+niPovhubp+j8cALnMAo/nAPAHASPpQDGQJZD+cIsDMC5DHEDA/
 & _
"R4xlwfwikgchM0GSJyHyFwokf2AoIcP5ymyQwvkwcxMk8NpMjODJXDmTZziyRQDBmcp9A
 & _
"F4QIlBzhBH+KgdQehLCpFWOobQzhVitHWBocwtRXDqEQBkLQdQPj7A8MsWQbBthOGiLU
 & _
"uH2LYeIvR3hbHcPMX47xOAGH6MAd7Lw4B3A+PwH4ExiBxC4G8DKwA+CPA6MkOQHBPgU
 & _
"wOgnCKNIecHRDB5A+P0L7Tw6jdEeGEbI9R+ivDKNsOwnRfhcECJkA4oR/jeHuP0e4gRwj4C
 & _
"fwfxRjoD0F4V4qR1j6D8M8Vo7Q9geG+KQQImgzjBH+PAfgfwLjJHmPwb4TxlyAA+Gcao9h+
```

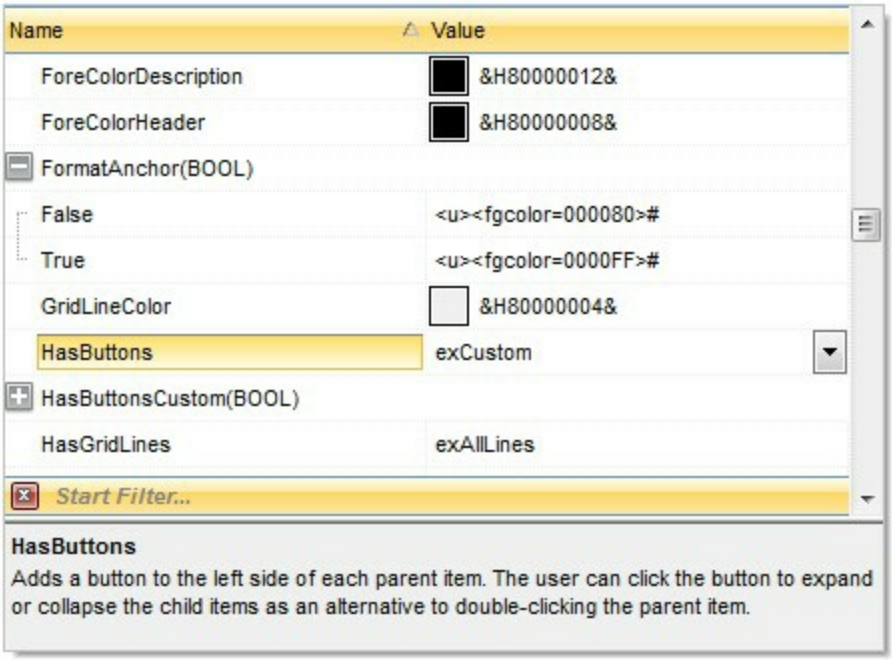
```
 & _
"+O8ekzhxj/D8MQBgbwHy4ACGAHgBhAgQBcAUIA2E/DgDIBAQI0AjAEKoMgH4hx+AXEGI
 & _
"A0CAM8bI+hChHHMBYIQmAtB9CQBkAIAiEFCIOF2jhRwsFJKWUwoAAQiUhKkVQqxWCtFcI
 & _
"dy8l7L6X8wJgzCiOGGGQWAOQCCKBGCgTgQgjAzEyHwawNwQCgCMNAagMACgNAaBcT
 & _
"YUQ0QRDRNwbo3hvjgHCOIJ4aoYhqgpFKD4boSRSjSFKdIOgvgxAAF2NAMo8w8hVHsJkP
 & _
"BcC0OAlRphpHkAsMAAwEgDFUEAeA1hsVXAyIoRY2AWgXEWA0TQyQxDTHIOoXIcADBXBu
 & _
"ag6GmFEVQ5g5DbESLQOYSwGjjEACwfQjQOBbD6FYXoawwg6GcCYHwqgECEEoA0EwMxC
 & _
"wEAMCLFQDkfYgRtD9GiPVNAvwPi/GuPceQ/xQBvGCMwPwHx4AcAWAIIA0AmARAKJ8d4x
 & _
"BofYQYEiwD4J+XQQwIBECiCwJIExhhnCIDoNAnhzj8CyBcIosQ+BlAwMZVAOgygeUMHEDoF
 & _
"IHkDQSIJRkhSDYISfIpxIj/BQMoOQIBUgrGUNIZgnh9gWGPGIFwyx5D0GCDEZgUwWC2DoE
 & _
"AOqGYCE4WYvB6g8GeJMHIQg7L0H6AMIAUAqAtASEMZg5BojUD+NEKgZQLhGGkBQPoJ
 & _
"c4/QghRCoDUDoTQpDVAmDkBgchPj1C6FsKwVRqiNDCFkZg4x8jGD+1AYobwuDXEqN0Po
 & _
"4Acfr6Qsi2D6KUNA2hJg4GsG8T49heizDZvQdouQ3jICUDUYocRuCXBaMYQIoAQEhKCUMA
 & _
"ldAwNQMwN6agjDDTgtRbC3FwLkXQuzLhEDIDUCUSQyRxAjCiIQUQEwWCXCMIAKIsASAm
 & _
"igCA2EIDgTIwQDDUEwJAbQeweCeEICEGoKr4gvBAAQIS2RJgZGgOoOQMhGTvFwBkBAmw
 & _
"4dg8heA6FeE4TY5R7JpGAGAN8YAqjhHKHYPo4RNCYFqP4GoeRHjIDIHILIPBTA/A0L8YotQY
 & _
"FONEM40BcjUFcBIfYqgxiwGwCwAo6h0iAF4NsNwNh9hOhzB3g+hdBtB2AsAdBWADBGhEE
 & _
"gHBLgkgxhjgvh0B4AEAWgJgOhEBhgshWAKhvAPgPhFhkB/h6A9A+BPBfgbhFheBhh3AQB9
 & _
"tsBfg4AFh9B+BfBTAVhxB2hdhDAwhKAyAKBygMghApAKASB4giAnhJhYBMgcB/hegIBaATA
 & _
```

```
"hQH1BSgAgcAmghgIg2AugLBigiBqAnAzBiQtAlA1ANAjBEgbAmAJMwA+gLgjgyBWA4A0E
 & _
"AkBJHpgRgLBlAZBAgUAkBkhZAogUgRhNBpAVB/AgBmADgEheA3BkhYhsgWAnAJgCBaBm
 & _
"A5gOBTgQgTAkAIhpAsgSgTRiA2gAAlBkhiBQgSgehIhphghkgdAXhWASB0A7tXgwgNAiHp
 & _
"hiAugTAhhpADgyB0BhAYBmgOgIgqhpAao/ArgrAKg2gyA7AaA2AZgIApArgCANgfBqArhK
 & _
"qRuguhCmifgMOjCpiqukqXumOnOoCyOpOqKdOrqeutBxOuOvOwOxOyOzO0O1O2O3O4
 & _
"PlPJArPKPLPMPNPOPPPQPRPSPTPUPVPWPXPYPZPaPbPcPdPePfPgPhPiPjPkPlPmPnPoPpPq
 & _
"P3P4P5P6P7P8P9P+P/QAQBQCQDQEQFQGQHQIQJQKQLQMQNQOQPQQQRQSQTQUQV
 & _
"AVBagqBUB9g/AfJnBzh3h+B+ggAHhMhFB7hdg3BwhMAygMgCgcgjGKhCgEgeAogJhyB2
 & _
"hlAggdhlAUgCA0AMhjA0ggRkAjh+GghihIxBgKhiByBqAkRFACAKAiRHRIRJRKA+JmBjxMxI
 & _
"glAZBOBQFpAZA2BKAlAkBzBaBFBfhlAZgEhhA3gNoSg7AlgJhCBggWgZhmhJgqBntSNTBN
 & _
"gOuwApA2gCAlBJA6AOgFghAoAJh0gOgegOBJhZB9BhUqA4htA7AIhSAkgHRwBSAsgRgm
 & _
"BmARgGNjh6g2gTgrgOhKhCgygrhrBDgxgjBhgYgCgSg6AghZh2ginQBaBGB+g6ApAYgEAE
 & _
"1cgZVYVXiEVZAAVaAagQAWgaViVjVkAEiEVe1gVaAbVj1oVkCAg=="
End With
```

If running the empty control we get the following picture:



If running the control using the code being generated by the VisualAppearance designer we get:



Property object

The Property object represents an object property, a category item, a property object page or a variable. Use the SelectedProperty to get the selected property. Use the IncludeProperty event to filter the object properties. The Property object supports the following properties and methods

Name	Description
AddValue	Adds a new item, when the property is of EnumType.
BackColor	Specifies the property's background color.
Bold	Specifies a value whether the property appears as bold.
Caption	Retrieves or sets the value's description in the predefined list.
Category	Retrieves a value indicating whether the item hosts a category.
CategoryName	Retrieves the property's category name.
CellBackColor	Specifies the cell's background color.
CellBackgroundExt	Indicates additional colors, text, images that can be displayed on the cell's background using the EBN string format.
CellBackgroundExtValue	Specifies at runtime, the value of the giving property for specified part of the background extension.
CellForeColor	Specifies the cell's foreground color.
Clear	Clears the predefined list values.
Description	Specifies the property's description.
DisplayCaption	Retrieves the property's full name.
DisplayCheck	Specifies whether the property displays a check box.
DisplayColor	Specifies whether the property displays colors.
DisplayDate	Specifies whether the property displays dates.
DisplayFile	Specifies whether the property displays files.
DisplayFolder	Specifies whether the property displays folders.
DisplaySlider	Specifies whether the property displays a slider.
DisplayValue	Retrieves the property's display value. The value is displayed by the properties browser.
	Retrieves or sets a value that indicates the maximum

DropDownItems	number of visible rows in a drop-down list.
EditType	Specifies the type of the property's editor.
Enabled	Enables or disables a property.
Flags	Retrieves the property's flags. This is a combination of FUNCFLAGS
ForeColor	Specifies the property's foreground color.
Height	Specifies the height in pixels of the property.
HTMLName	Displays the name of the property using built-in HTML format.
HTMLValue	Displays the value of the property using built-in HTML format.
ID	Specifies the property's identifier
ItemCollection	Retrieves a value that indicates whether the property is an item of a collection.
Locked	Specifies whether the property can be edited.
Mask	Specifies the property's mask.
MaskChar	Specifies the property's masking character.
Name	Retrieves the property's name.
Numeric	Specifies whether the property is of numeric type.
NumericFloat	Specifies whether the property is of float type.
Object	Retrieves the owner object of the property.
Option	Specifies an option for the property's editor.
Parameter	Specifies whether the item holds a parameter of the parent property.
Parent	Retrieves the parent of the property in the properties browser. Nothing if the property has no parent.
Position	Retrieves or sets a value that indicates the item's position in the children list.
PropertyObject	Retrieves a value that indicates whether the property is an object or non object.
PropertyPage	Retrieves a value that indicates whether the property contains a property page.
ReadOnly	Retrieves a value that indicates whether the property is

	read-only.
RemoveValue	Removes an item, when the property is of EnumType.
Selectable	Specifies whether the user can select the property.
SingleLine	Specifies whether the property is shown using single or multiple lines.
SliderMax	Specifies the slider's maximum value.
SliderMin	Specifies the slider's minimum value.
SliderStep	Specifies a value that represents the proposed change in the slider control's position.
SliderTickFrequency	Returns or sets a value that indicates the ratio of ticks on the slider control.
SliderWidth	Specifies the width the property's slider.
Sortable	Specifies whether the item that hosts the property may change its position while sorting. An unsortable item does not change its position while sort is performed.
SortItems	Sorts the list of items in a drop down list editor.
SpinStep	Specifies a value that represents the proposed change in the up-down control's position.
ToolTip	Specifies the property's tooltip.
Type	Retrieves the property's type.
UserData	Gets or sets the user-definable data for the current object.
Value	Retrieves or sets the property's value.
Variable	Retrieves a value that indicates whether the property is a variable or a property.

method Property.AddValue (Value as Long, Description as String)

Adds a new item, when the property is of EditEnum type.

Type	Description
Value as Long	A long expression that indicates the item's value.
Description as String	A long expression that indicates the item's caption.

Use the AddValue method to add new value to the list of property of EditEnum, EditDropDown type. Use the RemoveValue method to remove values from the property's list values. Use the SortItems to sort the values by description. Use the Caption property of a predefined value at runtime. Use the Add method to insert new properties to the browser.

The following sample adds few values to a property of EditEnum type:

```
With PropertiesList1
   With .Add("Border", 0, EditEnum)
      .AddValue 0, "0 - None"
      .AddValue 1, "1 - Fixed"
   End With
   .Refresh
End With
```

property Property.BackColor as Color

Specifies the property's background color.

Type	Description
Color	A color expression that indicates the background color of the property

Use the BackColor property to apply a background color to a property. Use the ForeColor property to change the property's foreground color. Use the CellBackColor property to change the cell's background color.

property Property.Bold as Boolean

Specifies a value whether the property appears as bold.

Type	Description
Boolean	A boolean expression that indicates whether a property appears as bold.

Use the Bold property to bold a property. The value for the ShowObjects property is important for the Bold property before adding items. Changing the ShowObjects property has no effect after adding items.

By default,

- if the ShowObjects property is True, the Bold property is true for all parent items. A parent items contains at least a child item inserted using the Add method. For instance, if the ShowObjects is True, when adding a new item to a parent item, the Bold property of the parent item will be set on True.
- if the ShowObjects property is False, the Bold property is false, for all items.

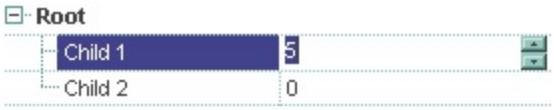
The following sample shows how to avoid bolding the parent items while building a hierarchy:

```
With PropertiesList1
   .BeginUpdate
   .ShowObjects = False
   .Add "Root", "", ReadOnly
   With .Add("Child 1", 0, Edit, , "Root")
      .Bold = True
   End With
   With .Add("Child 2", 0, Edit, , "Root")
      .Bold = True
   End With
   .ExpandItem("Root") = True
   .ShowObjects = True
   .EndUpdate
End With
```

In the sample, the root item is not bolded, instead the child items are bolded.

In the following sample the root item is automatically bolded, and the child items are not:

```
With PropertiesList1
   .BeginUpdate
   .ShowObjects = True
   .Add "Root", "", ReadOnly
   With .Add("Child 1", 0, Edit, , "Root")
   End With
   With .Add("Child 2", 0, Edit, , "Root")
   End With
   .ExpandItem("Root") = True
   .EndUpdate
End With
```



property Property.Caption(Value as Long) as String

Retrieves or sets the value's description in the predefined list.

Type	Description
Value as Long	A long expression that indicates the index of the item being changed.
String	A string expression that indicates the item's caption.

Use the Caption property to change the description for a predefined value. Use the AddValue method to add new items to an editor of drop down type. Use the Name property to retrieves the property's name.

property Property.Category as Boolean

Retrieves a value indicating whether the item hosts a category.

Type	Description
Boolean	A boolean expression that indicates whether the current object is a category object.

Use the CategoryName property to get the property's category name. Use the Category property to find whether the current object is a property item or a category item. Use the ShowCategories property to show the object properties. Use the Select method to browse an object. The Name property specifies the property's name. The Value property specifies the property's value.

property Property.CategoryName as String

Retrieves the property's category name.

Type	Description
String	A string expression that indicates the property's category name.

The ExPropertiesList control is able to categorize object properties. Use the ShowCategories property to enable categories into your control. Use the Category property to check if an item is a property or it is a category.

The following sample shows how to include only the properties in the "Misc" category:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not (Property.CategoryName = "Misc")
End Sub
```

property Property.CellBackColor([Index as Variant]) as Color

Specifies the cell's background color.

Type	Description
Index as Variant	A long expression that indicates the column's index. Valid values are: 0 for "Name" column, and 1 for "Value" column.
Color	A color expression that specifies the cell's background color.

Use the CellBackColor property to change the cell's foreground color. Use the BackColor property to change the foreground color for the entire property. Use the CellForeColor property to change the cell's foreground color.

The following sample changes the background/foreground color for the Border's value:

```
Private Sub Form_Load()
With PropertiesList1
   .BeginUpdate
      .Add "Appearance", "", ReadOnly
      With .Add("Border", 0, EditEnum, , "Appearance")
         .AddValue 0, "0 - None"
         .AddValue 1, "1 - Fixed"

         .CellBackColor(1) = vbRed
         .CellForeColor(1) = vbWhite
      End With
      .Refresh
   .EndUpdate
End With
End Sub
```

property Property.CellBackgroundExt(Column as Variant) as String

Indicates additional colors, text, images that can be displayed on the object's background using the EBN string format.

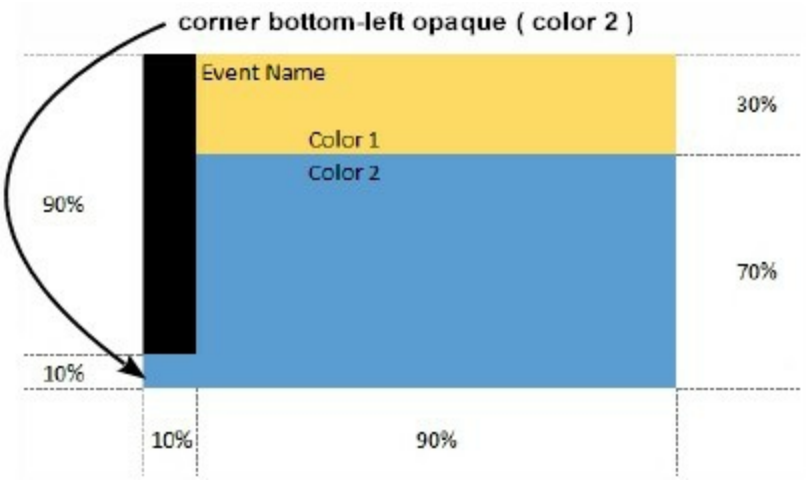
Type	Description
Column as Variant	A Long expression that specifies the index where the background extension is applied. The valid values are 0 for Name column, and 1 for Value column.
String	A String expression ("**EBN String Format**") that defines the layout of the UI to be applied on the object's 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.*

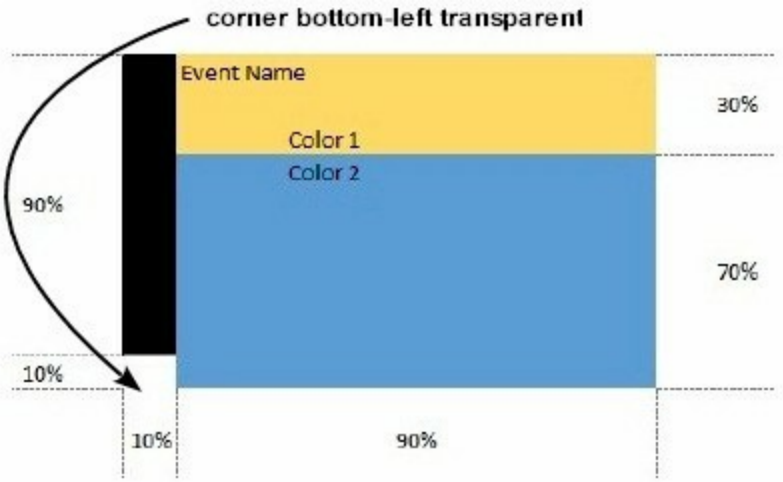
By default, the CellCellBackgroundExt property is empty. Using the CellCellBackgroundExt property you have unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the object's background. *For instance, let's say you need to display **more** colors on the object's 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 CellBackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The CellCellBackgroundExt property is applied right after setting the object's backcolor, and before drawing the default object's captions, icons or pictures.

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.

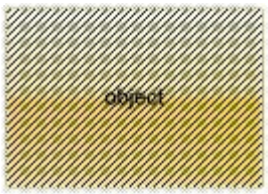
Complex samples:



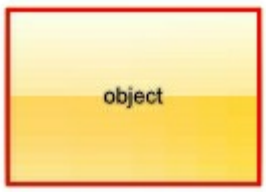


Easy samples:

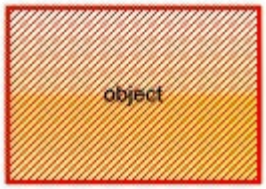
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



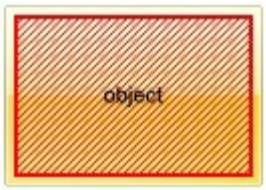
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



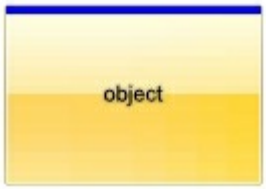
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



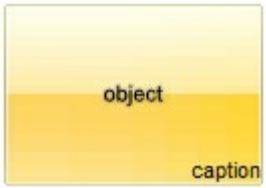
- "[[patterncolor=RGB(255,0,0)](none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.

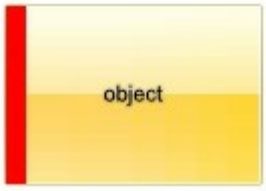


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

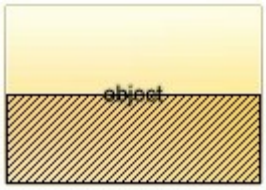


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

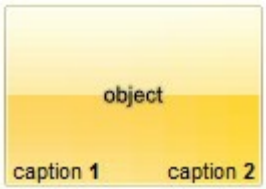
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



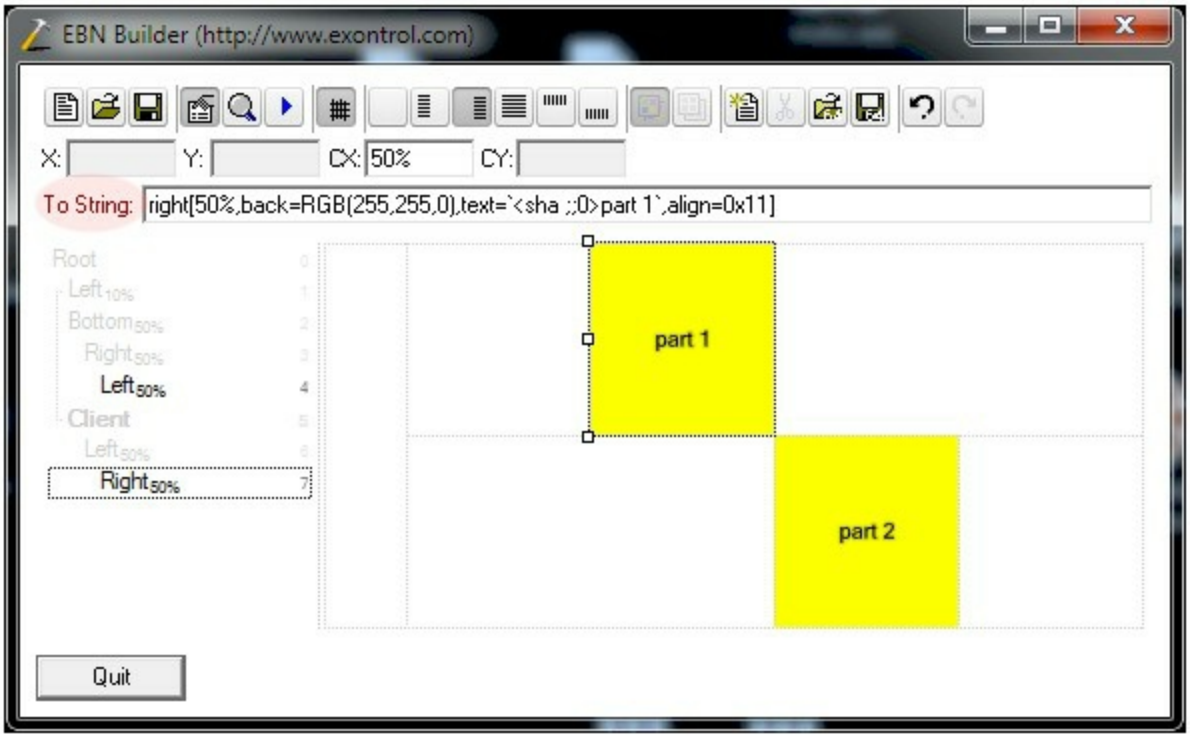
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



- "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:*



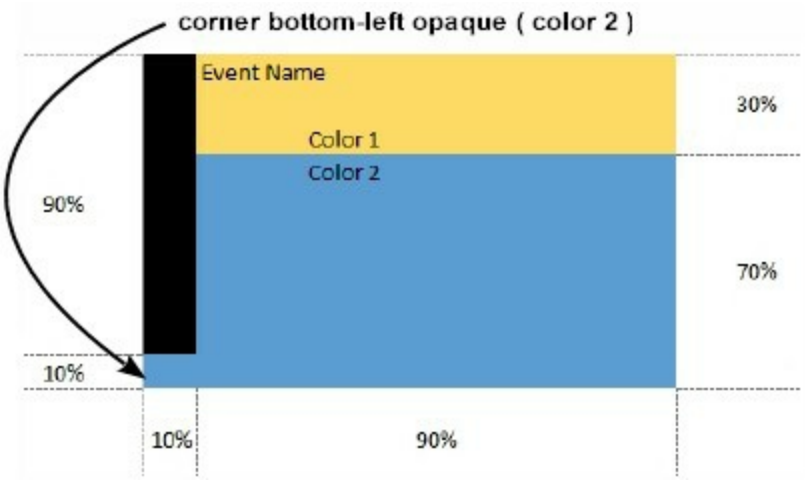
The **To String** field of the EBN Builder defines the **EBN String Format** that can be used on CellCellBackgroundExt property.

The **EBN String Format** syntax in BNF notation is defined like follows:

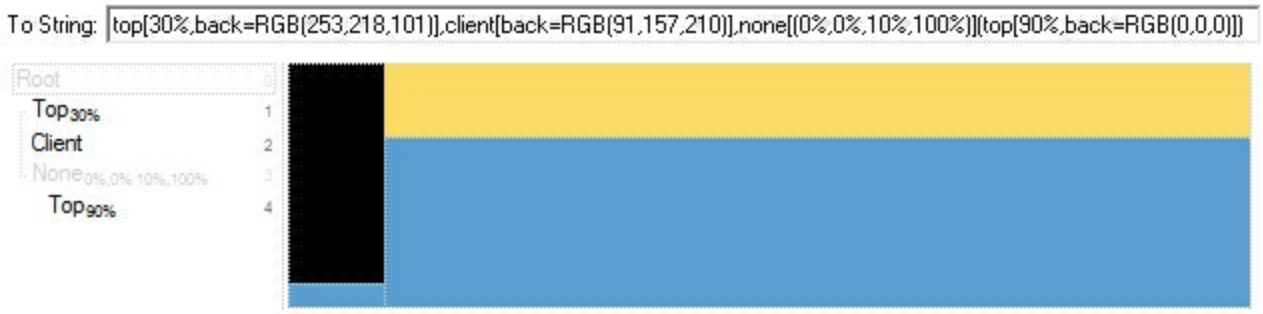
```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<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.

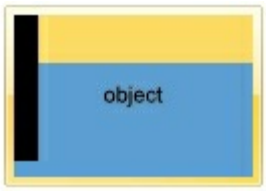
Now, lets say we have the following request to layout the colors on the objects:



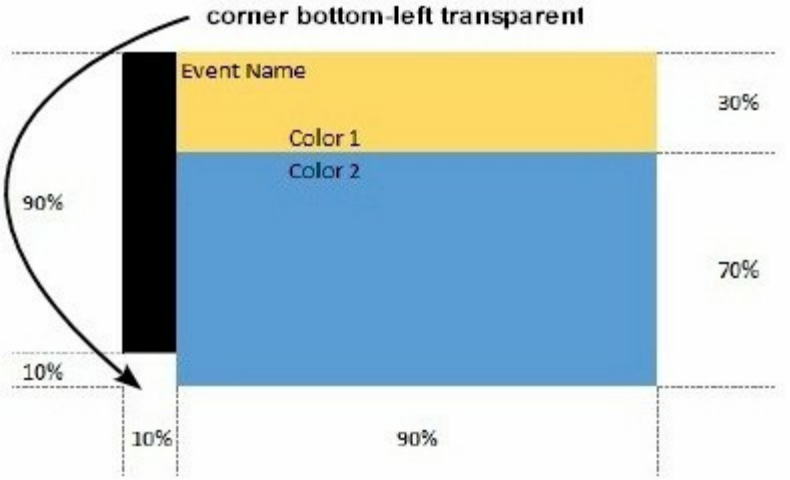
We define the CellCellBackgroundExt property such as "top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100%)](top[90%,back=RGB(0,0,0)])", and it looks as:



so, if we apply to our object we got:

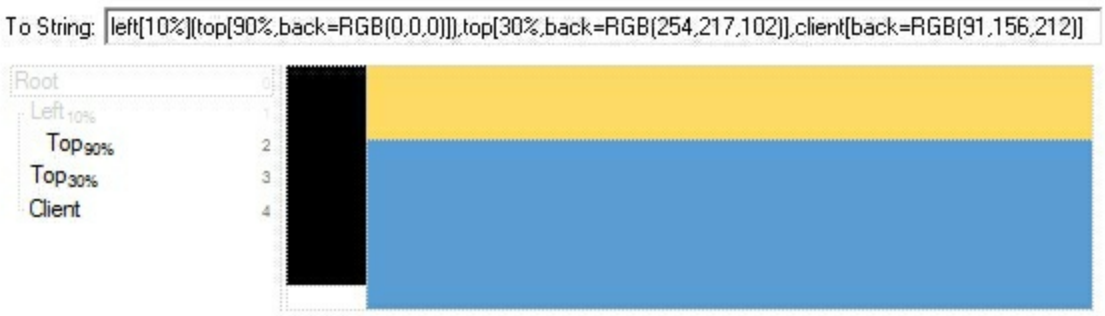


Now, lets say we have the following request to layout the colors on the objects:

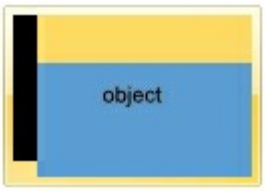


We define CellCellBackgroundExt property such as "left[10%]

(top[90%,back=RGB(0,0,0)]),top[30%,back=RGB(254,217,102)],client[back=RGB(91,156,2
and it looks as:

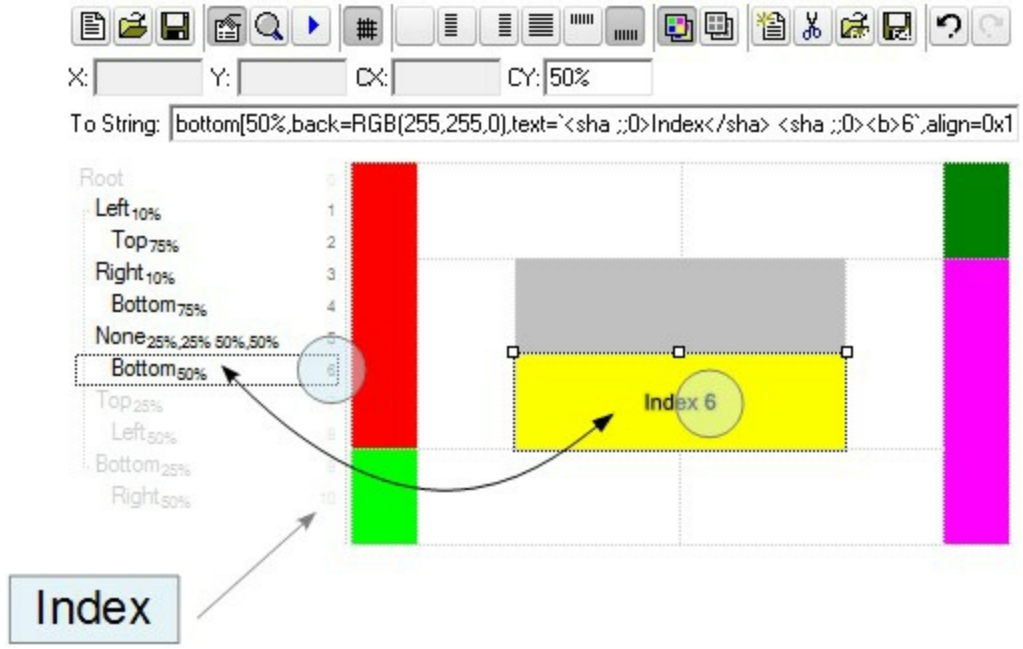


so, if we apply to our object we got:



property Property.CellBackgroundExtValue(Column as Variant, 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
Column as Variant	A Long expression that specifies the index where the background extension is applied. The valid values are 0 for Name column, and 1 for Value column.
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 CellBackgroundExt 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.*

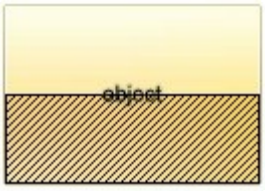
You can access/define/change the following UI properties of the element:

- **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)*

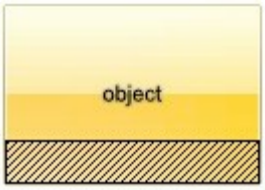
- **exFrameThickExt**(11), Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property. *(Boolean expression)*
- **exUserDataExt**(12), Specifies an extra-data associated with the object. *(Variant expression)*

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 Property.CellForeColor([Index as Variant]) as Color

Specifies the cell's foreground color.

Type	Description
Index as Variant	A long expression that indicates the column's index. Valid values are: 0 for "Name" column, and 1 for "Value" column.
Color	A color expression that specifies the cell's foreground color.

Use the CellForeColor property to change the cell's foreground color. Use the ForeColor property to change the foreground color for the entire property. Use the CellBackColor property to change the cell's background color.

The following sample changes the background/foreground color for the Border's value:

```
Private Sub Form_Load()
With PropertiesList1
   .BeginUpdate
      .Add "Appearance", "", ReadOnly
      With .Add("Border", 0, EditEnum, , "Appearance")
         .AddValue 0, "0 - None"
         .AddValue 1, "1 - Fixed"

         .CellBackColor(1) = vbRed
         .CellForeColor(1) = vbWhite
      End With
      .Refresh
   .EndUpdate
End With
End Sub
```

method Property.Clear ()

Clears the predefined list values.

Type	Description

Use the Clear method to clear all items of the property. Use the RemoveValue method to remove a particular value.

property Property.Description as String

Specifies the property's description.

Type	Description
String	A string expression that indicates the property's description.

Use the Description property to get the property's description. The property's description describes briefly what a property does. By default, the ExPropertiesList control displays in its description window the description for selected property. Use DescriptionVisible property to hide the control's description window. The Name property gets the property's name. The Value property gets the property's value.

The following sample shows how to implement by yourself a description window (to run the sample please include to your form a Label control):

```
Private Sub PropertiesList1_SelChange()
   Label1 = PropertiesList1.SelectedProperty.Description
End Sub
```

property Property.DisplayCaption (Mode as DisplayCaptionEnum) as Variant

Retrieves the property's full name.

Type	Description
Mode as DisplayCaptionEnum	A DisplayCaptionEnum expression that specifies the caption to retrieve.
Variant	A String/Variant expression that indicates the retrieved caption.

The DisplayCaption property gets the caption to be displayed in the Name or Value columns. The Name or HTMLName property specifies the caption to be displayed on the Name column. The Value property indicates the property's value, which is displayed on the Value column. The Description property specifies the property's description.

The DisplayCaption property gets:

- Gets the caption as displayed on Name column. The Name or HTMLName property specifies the caption to be displayed on the Name column.
- Gets the value as displayed on Value column. The Value property indicates the property's value, which is displayed on the Value column. For instance, if the property is of an enumeration type, the property displays its literals, instead the value of the property, so the DisplayCaption returns the literals.
- Gets the value as displayed on Description panel. The Description property specifies the property's description.

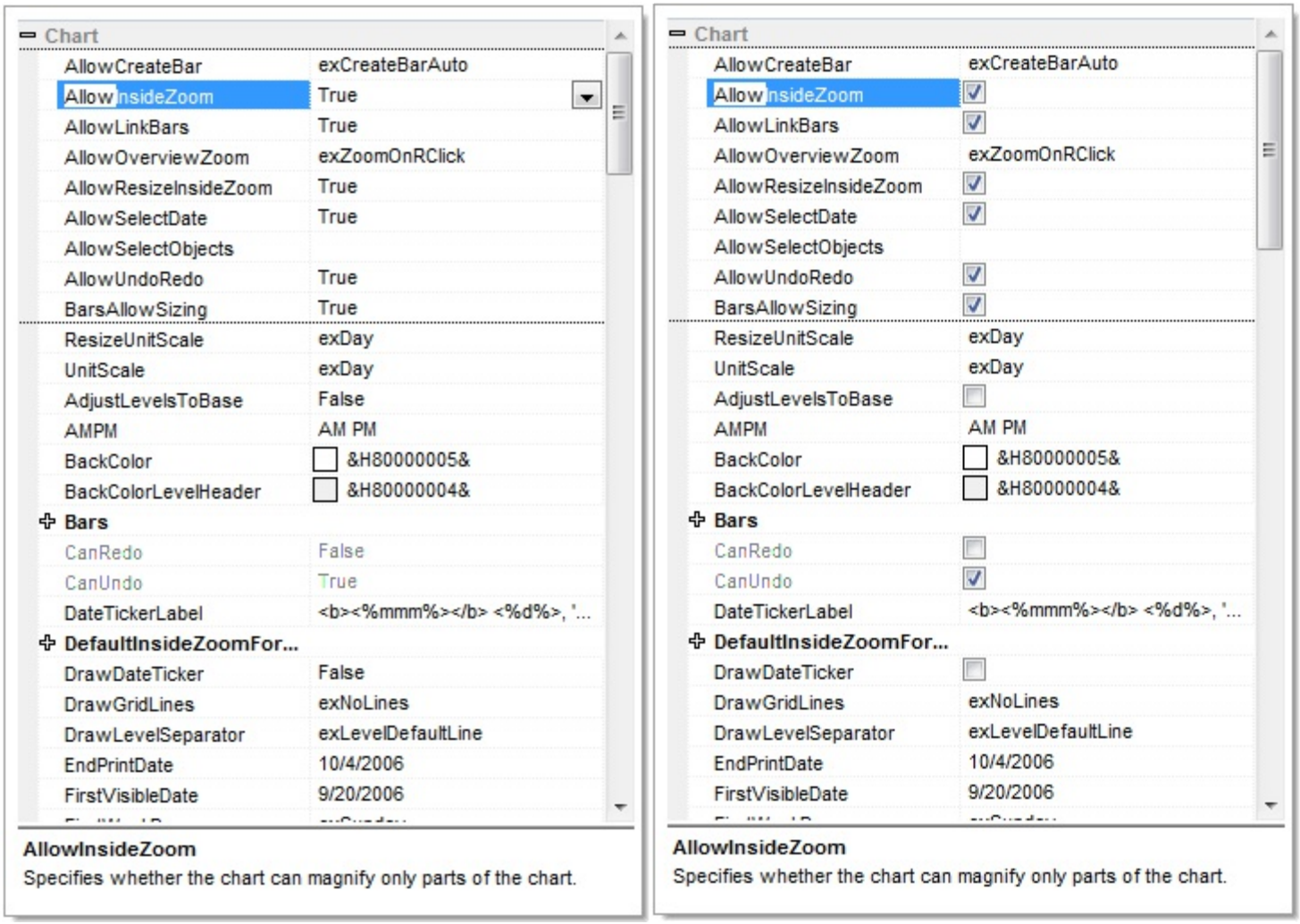
property Property.DisplayCheck as Boolean

Specifies whether the property displays a check box.

Type	Description
Boolean	A Boolean expression that indicates whether the property displays a check box.

By default, the DisplayCheck property is False. Use the DisplayCheck property to display a checkbox for boolean values. The PropertyChange event notifies your application that user is about to change a property. The control fires the PropertyChanged event when the user changes any property in the control. The DisplayColor, DisplayDate, DisplayFile or DisplaySlider property specifies whether the property displays a color, date, file, or slider.

For instance, the following screen shot lists the boolean properties on top, using True/False values, while the second screen shot shows the same properties displaying a checkbox instead True/False values.



The following VB sample changes the DisplayCheck property for properties of Boolean type:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    Property.DisplaySlider = InStr(1, Property.Name, "transparent", vbTextCompare) > 0
    If (Not Property.DisplaySlider) Then
        Property.DisplayDate = VarType(Property.Value) = vbDate
        If (Not Property.DisplayDate) Then
            Property.DisplayCheck = VarType(Property.Value) = vbBoolean
            If (Not Property.DisplayCheck) Then
                Property.DisplayColor = InStr(1, Property.Name, "color", vbTextCompare) > 0
            End If
        End If
    End If
End Sub
```

property Property.DisplayColor as Boolean

Specifies whether the property displays colors.

Type	Description
Boolean	A Boolean expression that indicates whether the property displays and edits a color value.

By default, the DisplayColor property is False. Use the DisplayColor property to let a property display and edit a color expression. For instance, let's say that your object lists a property of long type, but it changes the color for some part of the control. In this case, you can force the property to display the property as a color, and use the color editor to edit the value for the color. The PropertyChange event notifies your application that user is about to change a property. The control fires the PropertyChanged event when the user changes any property in the control. The DisplayCheck, DisplayDate, DisplayFile or DisplaySlider property specifies whether the property displays a checkbox, date, file, or slider.

property Property.DisplayDate as Boolean

Specifies whether the property displays dates.

Type	Description
Boolean	A Boolean expression that indicates whether the property displays and edits a date expression.

By default, the DisplayDate property is False. Use the DisplayDate property to let a property display and edit a date expression. For instance, let's say that your object lists a property of double type, but it changes the date. In this case, you can force the property to display the property as a date, and use the date editor to edit the value for the date. The PropertyChange event notifies your application that user is about to change a property. The control fires the PropertyChanged event when the user changes any property in the control. The DisplayCheck, DisplayColor, DisplayFile or DisplaySlider property specifies whether the property displays a checkbox, color, file, or slider.

property Property.DisplayFile as Boolean

Specifies whether the property displays files.

Type	Description
Boolean	A Boolean expression that indicates whether the property displays a path to a file.

By default, the DisplayFile property is False. Use the DisplayFile property to let a property display and edit a string expression that represents the path to a file. For instance, let's say that your object lists a property of string type that changes the path to a file, so you can assign the open file editor to change the listed path. The PropertyChange event notifies your application that user is about to change a property. The control fires the PropertyChanged event when the user changes any property in the control. The DisplayCheck, DisplayColor, DisplaySlider or DisplayDate property specifies whether the property displays a checkbox, color, slider, or date.

property Property.DisplayFolder as Boolean

Specifies whether the property displays folders.

Type	Description
Boolean	A Boolean expression that indicates whether the property displays a path to a file.

By default, the DisplayFolder property is False. Use the DisplayFolder property to let a property display and edit a string expression that represents the path to a file. For instance, let's say that your object lists a property of string type that changes the path to a file, so you can assign the open file editor to change the listed path. The PropertyChange event notifies your application that user is about to change a property. The control fires the PropertyChanged event when the user changes any property in the control. The DisplayCheck, DisplayColor, DisplaySlider or DisplayDate property specifies whether the property displays a checkbox, color, slider, or date.

property Property.DisplaySlider as Boolean

Specifies whether the property displays a slider.

Type	Description
Boolean	A Boolean expression that indicates whether the property displays a slider control.

By default, the DisplaySlider property is False. Use the DisplaySlider property to let a property display and edit a long expression between 0 and 100. For instance, let's say that your object lists a property Transparency of long type and so, you can assign a slider control to display and change the transparency for the object. The PropertyChange event notifies your application that user is about to change a property. The control fires the PropertyChanged event when the user changes any property in the control. The DisplayCheck, DisplayColor, DisplayFile or DisplayDate property specifies whether the property displays a checkbox, color, file, or date.

property Property.DisplayValue as String

Retrieves the property's display value. The value is displayed by the properties browser.

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

The control uses the DisplayValue property to display the property's value into the browser. To get the value of the property you should use the Value property. If the Category property is True, the DisplayValue gets the category name. For instance, a property of Color type, has the DisplayValue like &H80000005&, since the property's value is 0x80000005. A property of boolean type has DisplayValue like "True" or "False". The following sample shows how to print the property''s display value when selection is changing:

```
Private Sub PropertiesList1_SelChange()
   Debug.Print PropertiesList1.SelectedProperty.DisplayValue
End Sub
```

property Property.DropDownItems as Long

Retrieves or sets a value that indicates the maximum number of visible rows in a drop-down list.

Type	Description
Long	A long expression that indicates the number of visible rows in the drop down list editor.

By default, the DropDownRows property is 7. Use the DropDownRows property to specify number of visible items in a drop down list editor. The DropDownRows property has effect for editors like EditEnum, EditDropDown or EditFontName. The EditType property specifies the type of the editor assigned to a property.

property Property.EditType as EditTypeEnum

Specifies the type of the property's editor.

Type	Description
EditTypeEnum	An EditTypeEnum expression that indicates the type of the editor being used when the property is edited.

Use the Add method to add a custom property.

property Property.Enabled as Boolean

Enables or disables a property.

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

By default, the Enabled property is True. A disabled property is locked and it looks grayed. Use the Locked property to lock a property. A locked property doesn't look grayed.

property Property.Flags as Long

Retrieves the property's flags.

Type	Description
Long	A long expression that indicates the property's flags.

You can use the Flags property to filter only properties that have some flags, using the IncludeProperty event. Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

Here's the list of all flags:

Name	Value	Description
FUNCFLAG_FRESTRICTED	0x0001	The function should not be accessible from macro languages. This flag is intended for system-level functions or functions that type browsers should not display.
FUNCFLAG_FSOURCE	0x0002	The function returns an object that is a source of events.
FUNCFLAG_FBINDABLE	0x0004	The function that supports data binding.
FUNCFLAG_FREQUESTEDIT	0x0008	When set, any call to a method that sets the property results first in a call to IPropertyNotifySink::OnRequestEdit. The implementation of OnRequestEdit determines if the call is allowed to set the property.
FUNCFLAG_FDISPLAYBIND	0x0010	The function that is displayed to the user as bindable. FUNC_FBINDABLE must also be set.
FUNCFLAG_FDEFAULTBIND	0x0020	The function that best represents the object. Only one function in a type information can have this attribute.
FUNCFLAG_FHIDDEN	0x0040	The function should not be displayed to the user, although it exists and is bindable.
FUNCFLAG_USESGETLASTERROR	0x0080	The function supports GetLastError. If an error occurs during the function, the caller can call GetLastError to retrieve the error

		code.
FUNCFLAG_FDEFAULTCOLLELEM	0x0100	Permits an optimization in which the compiler looks for a member named ”xyz” on the type of ”abc”. If such a member is found and is flagged as an accessor function for an element of the default collection, then a call is generated to that member function. Permitted on members in dispinterfaces and interfaces; not permitted on modules.
FUNCFLAG_FUIDEFAULT	0x0200	The type information member is the default member for display in the user interface.
FUNCFLAG_FNONBROWSABLE	0x0400	The property appears in an object browser, but not in a properties browser.
FUNCFLAG_FREPLACEABLE	0x0800	Tags the interface as having default behaviors.
FUNCFLAG_FIMMEDIATEBIND	0x1000	Mapped as individual bindable properties

property Property.ForeColor as Color

Specifies the property's foreground color.

Type	Description
Color	A color expression that indicates the foreground color of the property.

Use the ForeColor property to apply a foreground color to a property. Use the BackColor property to change the property's background color. Use the CellForeColor property to change the cell's foreground color.

property Property.Height as Long

Specifies the height in pixels of the property.

Type	Description
Long	A long expression that indicates the height of a particular item/property.

Use the Height property to specify the height for a property. Use the DefaultItemHeight property to specify the height of the properties before loading properties to the control.Use the
 tag in HTMLName property to break a line.

property Property.HTMLName as String

Displays the name of the property using built-in HTML format.

Type	Description
String	A string expression that indicates the HTML format being displayed in the Name column, instead Name property.

By default, the HTMLName property is empty. If the HTMLName property is empty, the Name property is displayed in the name column. Use the HTMLName property to assign pictures, icons or font attributes to parts of the caption being displayed in the Name column. The HTMLValue property specifies the HTML caption to be displayed on the Value column. The SingleLine property indicates whether the property is displaying by single or multiple lines.

The HTMLName property 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.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on

the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

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

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

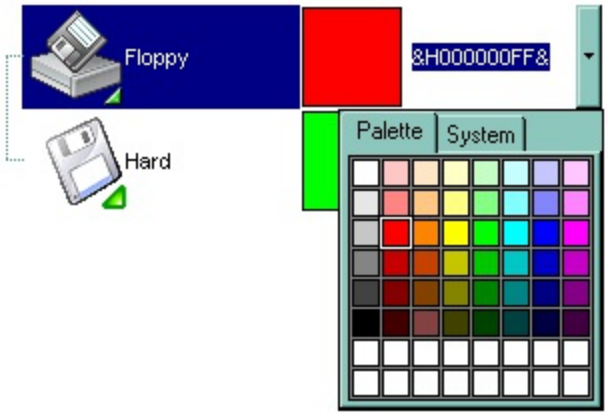
- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 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

For instance, the "<u>B</u>old" sequence displays a string like follows: **<u>B</u>old**. The control displays the HTMLName property instead Name property, only if the HTMLName is not empty. Use the IncludeProperty to assign a new html format for a specified property, like in the following sample.



For instance, the following VB sample adds an icon to the Appearance property.

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
```

```
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    If Property.Name Like "Appearance" Then
        Property.HTMLName = "<img>1</img>" & Property.Name
    End If
End Sub
```

For instance, the following template sample adds three properties and assign an icon to each of them, using the <img> tag.

```
BeginUpdate
Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl

Add("Icon1","",1).HTMLName = "1Icon 1"
Add("Icon2","",1).HTMLName = "2Icon 2"
Add("Icon3","",1).HTMLName = "3Icon 3"
EndUpdate
```

The following VB sample adds a picture to a property:

```
With PropertiesList1
    .HasGridLines = False
    .DefaultItemHeight = 52
    .HTMLPicture("floppy") = "D:\Temp\Icons\3floppy_1mount.gif"
    .HTMLPicture("hard") = "D:\Temp\Icons\3floppy_mount.gif"
    With .Add("Floppy", "", EditColor)
        .HTMLName = "<img>floppy</img>Floppy"
    End With
    With .Add("Hard", "", EditColor)
        .HTMLName = "<img>hard</img>Hard"
    End With
End With
```

property Property.HTMLValue as String

Displays the value of the property using built-in HTML format.

Type	Description
String	A String expression that defines the HTML caption to be displayed on the Value column.

By default, the HTMLValue property is empty. If HTMLValue property is empty, the Value property is displayed on the Value column. The SingleLine property indicates whether the property is displaying by single or multiple lines.

The HTMLValue property 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.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

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

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

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

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

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

property Property.ID as Long

Retrieves the property's identifier

Type	Description
Long	A long expression that indicates the property's identifier.

The property's identifier identifies a property. The property's identifier is used by Invoke method of IDispatch property to invoke a property. Use the Add method to specify the property's identifier when adding properties manually.

property Property.ItemCollection as Boolean

Retrieves a value that indicates whether the property is an item of a collection.

Type	Description
Boolean	A boolean expression that indicates whether the property is an item of a collection.

If the ItemCollection is True, the Property object contains an element of the collection. The ExPropertiesList control is able to display collections and their elements. When the control finds a property that exports a collection it adds the elements of the collection as child items of the property. During loading the control uses the NameItemsCollection property to determine the name of each element into collection, if it is a collection of objects. If the ItemCollection is True use the Object property to find the owner collection. For instance, if the collection contains only strings, the items added to browser's list will be numerated. Instead if the collection contains another objects, it uses the NameItemsCollection property to determine the caption that will be displayed on the name column. Let's suppose that we have the following collection:

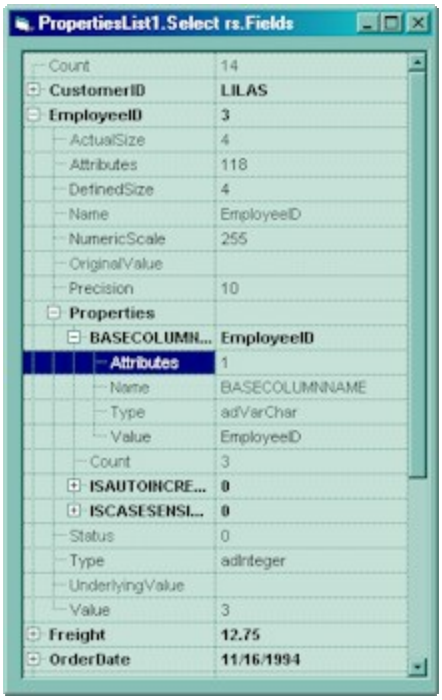
```
Dim n As New Collection
n.Add 10
n.Add 20
n.Add 30
```

When the control browses for the collection n (PropertiesList1.Select n), the elements of collection are not of object type, so the control will not be able to find each element name. Instead, if we have the following sample:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
```

and we call PropertiesList1.Select rs.Fields, the control will browse the rs.Fields collection. The Fields property of an ADO recordset contains a collection of Field objects. Since each Field object has a property called "**Name**", the browser will be able to find the element name using the NameItemsCollection property that contains by default: "**Name**;Caption;Item".

Here's a screen shot of how the ExPropertiesList browses the Fields collection (rs.Fields):



If we are using the following sample (make sure that you have set the NameItemsCollection to empty at design time):

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
PropertiesList1.Select rs.Fields
```

property Property.Locked as Boolean

Specifies whether the property can be edited.

Type	Description
Boolean	A boolean expression that specifies whether the property can be edited.

By default, the Locked property is False. Use the Locked property to lock a property. A locked property is not editable, and doesn't look grayed. Use the Enabled property to disable a property. A disabled property is locked and looks grayed. Use the ForeColor property to change the foreground color for a locked property.

Use the ReadOnly property to lock the entire control. Use the Enabled property to disable the entire control.

property Property.Mask as String

Specifies the property's mask.

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

he Mask property is composed by a combination of regular characters, literal escape characters, and masking characters. The Mask property can contain also alternative characters, or range rules. A literal escape character is preceded by a \ character, and it is used to display a character that is used in masking rules. Here's the list of all rules and masking characters:

Rule	Name	Description
#	Digit	Masks a digit character. [0-9]
x	Hexa Lower	Masks a lower hexa character. [0-9],[a-f]
X	Hexa Upper	Masks a upper hexa character. [0-9],[A-F]
A	AlphaNumeric	Masks a letter or a digit. [0-9], [a-z], [A-Z]
?	Alphabetic	Masks a letter. [a-z],[A-Z]
<	Alphabetic Lower	Masks a lower letter. [a-z]
>	Alphabetic Upper	Masks an upper letter. [A-Z]
*	Any	Mask any combination of characters.
\	Literal Escape	Displays any masking characters. The following combinations are valid: \#,\x,\X,\A,\?,\<,\>,\\,\{,\[
{nMin,nMax}	Range	Masks a number in a range. The nMin and nMax values should be numbers. For instance the mask {0,255} will mask any number between 0 and 255.
[...]	Alternative	Masks any characters that are contained by brackets []. For instance, the [abcA-C] mask any character: a,b,c,A,B,C

The following sample shows how to mask an IP address:

```
.Mask = "{0,255}\.{0,255}\.{0,255}\.{0,255}"
```

property Property.MaskChar as Long

Specifies the property's masking character.

Type	Description
Long	A value that indicates the character used for masking characters.

Use the MaskChar property to change the masking character. By default, the MaskChar is '_'.

property Property.Name as String

Retrieves the property's name.

Type	Description
String	A string expression that indicates the property's name.

If the Property object hosts a category the Name property retrieves the category name. Use the SelectedProperty event to find out the selected property. The Value property gets the property's value. The ID property gets the identifier of the property. The Description property gets the description of the property being displayed in the control's description bar. Use the UserData property to associate an extra data to the property. Use the HTMLName property to assign icons, pictures, font attributes or colors, to parts of the caption being displayed in the Name column.

The following sample prints the name and the type of the selected property (for instance, the sample is useful to find out the type of the property selected, when you need to include or exclude properties using the IncludeProperty event):

```
Private Sub PropertiesList1_SelChange()
    Debug.Print "You have selected the """ & PropertiesList1.SelectedProperty.Name & """.
The type for it is: " & PropertiesList1.SelectedProperty.Type
End Sub
```

property Property.Numeric as Boolean

Specifies whether the property is of numeric type.

Type	Description
Boolean	A boolean expression that indicates whether property's edit box allows only digits to be entered.

The Numeric property allows only digits to be entered into the property's edit box. Use the AllowSpin property to specify whether the numeric properties show a spin control. Use the SpinStep property to define the proposed change when user clicks the spin.

property Property.NumericFloat as Boolean

Specifies whether the property is of float type.

Type	Description
Boolean	A boolean expression that indicates whether the property is of float type.

A float type is represented in the format: **[+/-]digit[.]digit[e/E/d/D][+/-]digit**, where **digit** is any combination of digit characters. Use the Numeric property to specify when only long values are allowed in the property. Use the AllowSpin property to specify whether the numeric properties show a spin control. Use the SpinStep property to define the proposed change when user clicks the spin. If the NumericFloat property is True, the input characters are filtered, and so characters that are not in the number representation can't be inserted. For instance, if you have a number such of 100, and you type 'e' character, the number will be 100e, but if you would type again a new 'e' it will not be accepted, so the number will stay as 100e.

property Property.Object as Object

Retrieves the owner object of the property.

Type	Description
Object	An object that's the owner of the property.

The Object property retrieves the owner object of the property. Use the Value property to get the object contained by the property. For instance if a Property object has the ItemCollection property to True, then the Object property retrieves the owner object collection. Also, if a Property object contains is of Variable type, the Object property retrieves the object where the variable belong. Use the PropertyObject property to determine whether the property contains an object. For instance a property of font type is considered as been of object type. You can use the Type property to determine the type of the property. The following sample shows how to exclude variables of IPictureDisp properties:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   If Not (Property.Object Is Nothing) Then
      Cancel = Property.Variable And TypeOf Property.Object Is IPictureDisp
   End If
End Sub
```

property Property.Option(Name as OptionEnum) as Variant

Specifies an option for the property's editor.

Type	Description
Name as OptionEnum	An OptionEnum expression that indicates the option being changed.
Variant	A Variant value that indicates the option's newly value.

Use the Option property to change a particular option for a specified entry/property. If no option is specified for the current entry, the Option property of the control indicates the default options. The Option property applies the options to all editors, while the Option property of Property object may specify different options for different entries in the control. For instance, you can display a filter for some EditFile entries, and other filters for other EditFile entries in the same control.

In conclusion, you can specify options for the editors as follows:

- the same settings for all editors using Option property (by default).
- custom settings for the editor of an entry/property using the Property.Option property

The following VB sample adds a custom EditDate editor that uses Romanian calendar:

```
With PropertiesList1
   With .Add("Date", Date, EditDate)
   .Option(exDateTodayCaption) = "Azi"
   .Option(exDateMonths) = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August
Septembrie Octombrie Decembrie"
   .Option(exDateWeekDays) = "D L M M J V S"
   .Option(exDateFirstWeekDay) = 1
   Wnd With
End With
```

The following VB sample adds a custom EditFile editor with INI filter:

```
With PropertiesList1
   With .Add("INI", "c:\temp\test.ini", EditFile, "Selects a file", "Custom")
   .Option(exEditFileFilter) = "INI Files|*.ini;*.init|All (*.*)|*.*"
   .Option(exEditFileTitle) = "Select an INI file"
   End With
End With
```

property Property.Parameter as Boolean

Specifies whether the item holds a parameter of the parent property.

Type	Description
Boolean	A Boolean expression that indicates whether the property holds a parameter

The control may browse properties with multiple parameters.

property Property.Parent as Property

Retrieves the parent of the property in the properties browser.

Type	Description
Property	A Property object that's the parent of the property in the properties browser.

If the property has no parent, the Parent retrieves nothing.

property Property.Position as Long

Retrieves or sets a value that indicates the item's position in the children list.

Type	Description
Long	A long expression that indicates the position of the Property in the list of children items. 0 means the first property, 1, the second, and so on.

The Position property specifies the position of the property in the browser. Use the Position property to change the property's position. For instance, if the IncrementalSearch property is exContains + exMoveOnTop, the items are re-arranged so, the first items contain the typed characters, while the rest stay unchanged.

property Property.PropertyObject as Boolean

Retrieves a value that indicates whether the property is an object or non object.

Type	Description
Boolean	A boolean expression that indicates whether the Property object contains an Object.

Use the Value property to get the object contained by the property. All the properties of object type are displayed as bold. If the Property object contains a category item, the PropertyObject property is False. Use the ShowObjects property to show properties of object type into the control. For instance a property of IFontDisp type is considered as been as object type. Use the Type to get the string representation of the property's object type. The following sample shows how to simulate the VB browser (make sure that ShowObjects property is True):

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Property.PropertyObject
   If (Cancel) Then
      Cancel = Not (Property.Type = "Font*" Or Property.Type = "Picture*")
   End If
End Sub
```

property Property.PropertyPage as Boolean

Retrieves a value that indicates whether the Property object contains a property page.

Type	Description
Boolean	A boolean expression that indicates whether the Property object contains a property page.

The ExPropertiesList control supports browsing property pages. The PropertPage property specifies whether the Property object contains a property page. The following sample shows how to include into your browser only the property pages:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    Cancel = Not Property.PropertyPage
End Sub
```

property Property.ReadOnly as Boolean

Retrieves a value that indicates whether the property is read-only.

Type	Description
Boolean	A boolean expression that indicates whether the property is read-only.

The control displays the read-only properties in a grayed color. Use the ReadOnly property to disable user editing. Use the ShowReadOnly property to ignore read-only properties. If the item is of category type or property page type, the ReadOnly property has no effect. Use the Enabled property to disables a property. Use the Locked property to lock a property.

method Property.RemoveValue (Value as Long)

Removes an item, when the property is of EnumType.

Type	Description
Value as Long	A long expression that indicates the value of the item being removed.

Use the RemoveValue method to remove an item from the property's list values when it is of EnumType type. Use the AddValue property to add new values to property's list values.

property Property.Selectable as Boolean

Specifies whether the user can select the property.

Type	Description
Boolean	A Boolean expression that specifies whether the user can select the property at runtime.

By default, the Selectable property is True. Use the Selectable property to prevent a property to be selected. The Sortable property indicates whether the property's position is changed once the user sorts a column. The SelectedProperty property indicates the currently selected property. The control fires the SelChange event once a new property gets selected in the properties browser control.

property Property.SingleLine as Boolean

Specifies whether the property is shown using single or multiple lines.

Type	Description
Boolean	A Boolean expression that specifies whether the property is displaying by single or multiple lines.

By default, the SingleLine property is True. Use the SingleLine property to display the property's name or value on multiple lines. Use the Name or HTMLName property to specify the caption of the property to be displayed on the Name column. Use the Value or HTMLValue property to specify the caption of the property to be displayed on the Value column.

property Property.SliderMax as Double

Specifies the slider's maximum value.

Type	Description
Double	A double expression that indicates the maximum value for the slider control.

By default, the SliderMax property is 100. Use the EditSlider type when adding the property to assign a slider control to a property. The SliderWidth property determines the width of the slider in the property. The SliderStep property determines the slider step, when user moves the slider. The SliderMin and SliderMax properties determine the range being used by the Value property. The SliderTickFrequency property specifies the frequency to display ticks on a slider control.

property Property.SliderMin as Double

Specifies the slider's minimum value.

Type	Description
Double	A double expression that indicates the minimum value for the slider.

By default, the SliderMin property is 0. Use the EditSlider type to assign a slider to a property. The SliderWidth property determines the width of the slider in the property. The SliderStep property determines the slider step, when user moves the slider. The SliderMin and SliderMax properties determine the range being used by the Value property. The SliderTickFrequency property specifies the frequency to display ticks on a slider control.

property Property.SliderStep as Double

Specifies a value that represents the proposed change in the slider control's position.

Type	Description
Double	A double expression that indicates the slider's step.

The SliderStep property determines the slider step, when user moves the slider. Use the EditSlider type to assign a slider to a property. The SliderWidth property determines the width of the slider in the property. The SliderMin and SliderMax properties determine the range being used by the Value property. By default, the SliderStep property is 1.

property Property.SliderTickFrequency as Double

Returns or sets a value that indicates the ratio of ticks on the slider control.

Type	Description
Double	A Double expression that indicates the ratio of ticks on the slider control.

By default, the SliderTickFrequency property is 0. If the SliderTickFrequency property is 0 the slider displays no ticks. The SliderTickFrequency property specifies the frequency to display ticks on a slider control. Use the EditSlider type when adding the property to assign a slider control to a property. The SliderWidth property determines the width of the slider in the property. The SliderStep property determines the slider step, when user moves the slider. The SliderMin and SliderMax properties determine the range being used by the Value property.

property Property.SliderWidth as Long

Specifies the width the property's slider.

Type	Description
Long	A long expression that indicates the width of the slider in the property like explained bellow.

Use the EditSlider type to assign a slider to a property. By default, the SliderWidth property is 64 pixels.

- If the SliderWidth property is 0, the slider control is not visible.
- If the SliderWidth property is greater than 0, the SliderWidth property represents the width in pixels of the slider in the control.
- If the SliderWidth property is less than 0, the absolute value of the SliderWidth property represents the percent being used by the slider in the property. For instance, if the SliderWidth property is -50, that means the that slider's width will be 50% (half) of the cell's width.

Use the SliderMin and SliderMax properties to hold the property's Value within a range. The SliderStep property determines the slider step, when user moves the slider. The SliderTickFrequency property specifies the frequency to display ticks on a slider control.

The following sample adds different types of sliders:

```
With PropertiesList1
  .AllowSpin = True
  With .Add("Spin", 0.2, Edit)
    .NumericFloat = True
  End With
  With .Add("Slider with a non fixed width", 0.2, EditSlider)
    .SliderWidth = -60
    .SliderStep = 0.05
    .SpinStep = 0.05
    .SliderMin = 0
    .SliderMax = 50
  End With
  With .Add("Slider with a fixed width", 0, EditSlider)
    .SliderMin = -50
    .SliderMax = 50
  End With
```

```
    With .Add("Slider with unknown step", 0, EditSlider)
        .SliderWidth = -70
        .SpinStep = 0
        .SliderStep = 0
        .SliderMin = -50
        .SliderMax = 50
    End With
End With
```

property Property.Sortable as Boolean

Specifies whether the item that hosts the property may change its position while sorting.

Type	Description
Boolean	A Boolean expression that specifies whether the property is sortable or not.

By default, the Sortable property is True. An unsortable item does not change its position while sort is performed. The Sort method sorts the properties. Use the SortObjects property to specify if the object properties should be placed on top or bottom side of the control once the user sorts a column. Use the SortOnClick property to specify whether a column gets sorted once the user clicks the column's header.

method Property.SortItems ([Ascending as Variant], [Reserved as Variant])

Sorts the list of items in a drop down list editor.

Type	Description
Ascending as Variant	A boolean expression that indicates the sort order of the items. By default, is the Ascending parameter is True, if it is missing.
Reserved as Variant	Not used. For future use only.

Use the SortItems method to sort the items in a drop down list editor. Use the AddValue method to add predefined values to the drop down list. Call the SortItems after adding values to a drop down list editor.

property Property.SpinStep as Double

Specifies a value that represents the proposed change in the up-down control's position.

Type	Description
Double	A double expression that indicates the proposed change in the spin control.

Use the AllowSpin property to associate a spin (up/down control) to a property of numeric type. Use the SpinStep property to specify the proposed change when user clicks a spin control. By default, the SpinStep property is 1. If the SpinStep property is 0, the property doesn't show a spin control even if the property is of numeric type and AllowSpin property is True. The SliderTickFrequency property specifies the frequency to display ticks on a slider control.

property Property.ToolTip as String

Specifies the property's tooltip.

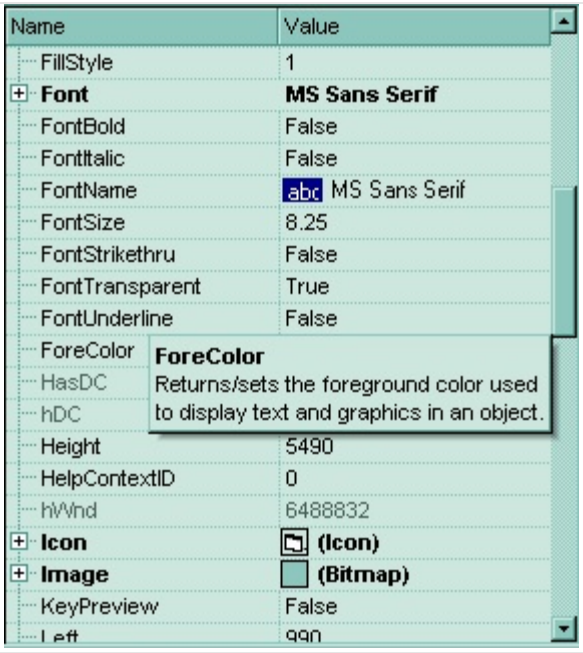
Type	Description
String	A string expression that indicates the property's tooltip.

Use the ToolTip property to assign a custom tooltip to a property. By default, the ToolTip property is "...". The AllowToolTip property specifies whether the control displays a tooltip when the string value is too long to be displayed in the property's client area. The control pops up a tooltip when the mouse pointer hovers the property's name or value in the following cases:

- If the ToolTip property is "...", and the property's name or property's value is too long to be displayed in the property's client area.
- The ToolTip property is not empty, and it is different than "..." (three dots).

The following sample shows how to display the property's description when the mouse pointer hovers the property's name:

```
Private Sub Form_Load()
   With PropertiesList1
      .DescriptionVisible = False
      .AllowTooltip = True
      .Select Me
   End With
End Sub

Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Property.ToolTip = Property.Description
End Sub
```


Name
Value
FillStyle
1
Font
MS Sans Serif
FontBold
False
FontItalic
False
FontName
abc MS Sans Serif
FontSize
8.25
FontStrikethru
False
FontTransparent
True
FontUnderline
False
ForeColor
HasDC
hDC
ForeColor
Returns/sets the foreground color used to display text and graphics in an object.
Height
5490
HelpContextID
0
hWnd
6488832
Icon
(Icon)
Image
(Bitmap)
KeyPreview
False
Left
990

property Property.Type as String

Retrieves the property's type.

Type	Description
String	A string expression that represents the property's type.

Use the PropertyObject property to check whether a property is of object type. For instance, the Type property is useful to determine whether a variable is member of a IFontDisp object, using the following statement: typeof Property.Object is IFontDisp. Use the Value property to determine the property's value. It is not recommended using the typeof Property.Value is IFontDisp, because the VB operator typeof is not able to check the type objects that are set to nothing. Use the SelectedProperty event to find out the selected property. The following sample prints the name and the type of the selected property (for instance, the sample is useful to find out the type of the property selected, when you need to include or exclude properties using the IncludeProperty event):

```
Private Sub PropertiesList1_SelChange()
   Debug.Print "You have selected the """ & PropertiesList1.SelectedProperty.Name & """.
The type for it is: " & PropertiesList1.SelectedProperty.Type
End Sub
```

property Property.UserData as Variant

Gets or sets the user-definable data for the current object.

Type	Description
Variant	A Variant value that defines the property's user data.

Use the UserData event to associate a user value to a property. Use the Property property to access to a property object. The Name property specifies the property's name. The Value property specifies the property's value. Use the ID property to identify a property.

The following sample associates an extra string to the property "Visible":

```
PropertiesList1.Property("Visible").UserData = "A constant string"
```

property Property.Value as Variant

Retrieves or sets the property's value

Type	Description
Variant	A variant expression that indicates the property's value.

If the property object holds a property page or a category, the Value property has no sense. Use the PropertyObject property to check whether the property is of object type. Use the value property to determine the object contained by the property. Use the Object to get the owner object of a property. Use the DisplayValue property to get the text that control displays in the browser. The control fires the EditChange event while user types characters in the property's text box control. The Name property retrieves the name of the property. Use the AddValue method to add predefined values to a drop down list editor. The DisplayValue property gets the string being displayed in the browser. The HTMLValue property specifies the HTML caption to be displayed on the Value column.

If the property is added manually, using the Add method, you need to call the Value property each time when you need to refresh the property's value. In case the properties list browses a COM object, using the Select method, you need to call the Refresh method to refresh the values for the properties in the browser, in case some changes occurs to the browsed object or if you need to. Also, if a property contains another COM object (EditObject type), the Refresh method updates the values for all browsed properties.

property Property.Variable as Boolean

Retrieves a value indicating whether the property is a variable.

Type	Description
Boolean	A boollean expression indicating whether the property is a variable.

If the Variable is False, that doesn't means that the Property object contains a real property. Use the PropertyPage property to determine if the Property object contains a properties page, or use the Category to check whether the Property object contains a category. Use the ShowVariables property to exclude variables. Use the Variable property to filter your items like in the following sample:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   If Not (Property.Object Is Nothing) Then
      Cancel = Property.Variable And TypeOf Property.Object Is IPictureDisp
   End If
End Sub
```

ExPropertiesList events

The ExPropertiesList component supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user presses and then releases the mouse button over the control.
DblClick	Occurs when the user dblclk the left mouse button over an object.
EditChange	Fired when user alters the text of an edit control.
Event	Notifies the application once the control fires an event.
IncludeProperty	Fired when the properties browser is about to include a new property.
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.
ModalPropertyChange	Fired when the properties browser is about to change a property's value using a modal dialog.
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.
PropertyChange	Fired when the properties browser is about to change a property's value.
PropertyChanged	Occurs after the property's value is changed.
ScrollButtonClick	Occurs when the user clicks a button in the scrollbar.
SelChange	Fired when the selected property is changed.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Type	Description
AnchorID as String	A string expression that indicates the identifier of the anchor
Options as String	A string expression that specifies options of the anchor element.

The control fires the AnchorClick event to notify that the user clicks an anchor element. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The AnchorClick event is fired only if prior clicking the control it shows the hand cursor. For instance, if the cell is disabled, the hand cursor is not shown when hovers the anchor element, and so the AnchorClick event is not fired. Use the FormatAnchor property to specify the visual effect for anchor elements. For instance, if the user clicks the anchor *<a1>anchor*, the control fires the AnchorClick event, where the AnchorID parameter is 1, and the Options parameter is empty. Also, if the user clicks the anchor *<a 1;yourextradata>anchor*, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "yourextradata". Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor.

Syntax for AnchorClick event, **/NET** version, on:

C#
```
private void AnchorClick(object sender,string AnchorID,string Options)
{
}
```

VB
```
Private Sub AnchorClick(ByVal sender As System.Object,ByVal AnchorID As
String,ByVal Options As String) Handles AnchorClick
End Sub
```

Syntax for AnchorClick event, **/COM** version, on:

C#
```
private void AnchorClick(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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 (Property as Property, Button as Integer, Shift as Integer)

Occurs when the user presses and then releases the mouse button over the control.

Type	Description
Property as Property	A Property object that indicates the property being clicked.
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.

The Click event is fired when user clicks a property. The Property parameter gets the property being clicked. The Value property gets the property's value. The Name property gets the property's name.

Syntax for Click event, **/NET** version, on:

C#

```
private void Click(object sender,exontrol.EXPROPERTIESLISTLib.Property Property,short Button,short Shift)
{
}
```

VB

```
Private Sub Click(ByVal sender As System.Object,ByVal Property As exontrol.EXPROPERTIESLISTLib.Property,ByVal Button As Short,ByVal Shift As Short) Handles Click
End Sub
```

Syntax for Click event, **/COM** version, on:

C#

```
private void ClickEvent(object sender, AxEXPROPERTIESLISTLib._IPropertiesListEvents_ClickEvent e)
{
}
```

C++

```
void OnClick(LPDISPATCH Property,short Button,short Shift)
{
}
```

C++ Builder

```
void __fastcall Click(TObject *Sender,Expropertieslistlib_tlb::IProperty *Property,short
Button,short Shift)
{
}
```

Delphi

```
procedure Click(ASender: TObject; Property : IProperty;Button : Smallint;Shift :
Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ClickEvent(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ClickEvent);
begin
end;
```

Powe…

```
begin event Click(oleobject Property,integer Button,integer Shift)
end event Click
```

VB.NET

```
Private Sub ClickEvent(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ClickEvent) Handles ClickEvent
End Sub
```

VB6

```
Private Sub Click(Property As EXPROPERTIESLISTLibCtl.IProperty,Button As
Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub Click(ByVal Property As Object,ByVal Button As Integer,ByVal Shift As
Integer)
End Sub
```

VFP

```
LPARAMETERS Property,Button,Shift

```

Xbas…

```
PROCEDURE OnClick(oPropertiesList,Property,Button,Shift)
RETURN
```

Syntax for Click event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="Click(Property,Button,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function Click(Property,Button,Shift)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComClick Variant llProperty Short llButton Short llShift
    Forward Send OnComClick llProperty llButton llShift
End_Procedure
```

Visual Objects
```
METHOD OCX_Click(Property,Button,Shift) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_Click(COM _Property,int _Button,int _Shift)
{
}
```

XBasic
```
function Click as v (Property as OLE::Exontrol.PropertiesList.1::IProperty,Button as N,Shift as N)
end function
```

dBASE
```
function nativeObject_Click(Property,Button,Shift)
return
```

The following sample prints the property's name if the user clicks the property:

```
Private Sub PropertiesList1_Click(Property As EXPROPERTIESLISTLibCtl.IProperty, Button As Integer, Shift As Integer)
    Debug.Print Property.Name
End Sub
```

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.

The DblClick event is fired when user double clicks the control.

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,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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.PropertiesList.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.PropertiesList.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_DblClick(Shift,X,Y)
return
```

event EditChange (Property as Property, Value as Variant)

Fired when user alters the text of an edit control.

Type	Description
Property as Property	A Property object being changed.
Value as Variant	A string expression that indicates the caption of the text box control.

The EditChange event is fired when user alters the text of the control's edit box. The PropertyChange event is fired when user changes the property's value. Use the Value property to access the old value for the property being changed. Use the ID property to identify a property. The KeyDown event is fired when user presses a key. The KeyPress event occurs when the user presses and releases an ANSI key. The control fires the SelChange event when a new property is selected. Use the UserData property to associate an extra data to a property.

Syntax for EditChange event, **/NET** version, on:

C#
```
private void EditChange(object sender,exontrol.EXPROPERTIESLISTLib.Property Property,ref object Value)
{
}
```

VB
```
Private Sub EditChange(ByVal sender As System.Object,ByVal Property As exontrol.EXPROPERTIESLISTLib.Property,ByRef Value As Object) Handles EditChange
End Sub
```

Syntax for EditChange event, **/COM** version, on:

C#
```
private void EditChange(object sender, AxEXPROPERTIESLISTLib._IPropertiesListEvents_EditChangeEvent e)
{
}
```

C++
```
void OnEditChange(LPDISPATCH Property,VARIANT FAR* Value)
{
}
```

C++ Builder

```
void __fastcall EditChange(TObject *Sender,Expropertieslistlib_tlb::IProperty
*Property,Variant * Value)
{
}
```

Delphi

```
procedure EditChange(ASender: TObject; Property : IProperty;var Value :
OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure EditChange(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_EditChangeEvent);
begin
end;
```

Powe…

```
begin event EditChange(oleobject Property,any Value)
end event EditChange
```

VB.NET

```
Private Sub EditChange(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_EditChangeEvent) Handles
EditChange
End Sub
```

VB6

```
Private Sub EditChange(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty,Value As Variant)
End Sub
```

VBA

```
Private Sub EditChange(ByVal Property As Object,Value As Variant)
End Sub
```

VFP

```
LPARAMETERS Property,Value
```

Xbas…

```
PROCEDURE OnEditChange(oPropertiesList,Property,Value)
RETURN
```

Syntax for EditChange event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="EditChange(Property,Value)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function EditChange(Property,Value)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComEditChange Variant llProperty Variant llValue
    Forward Send OnComEditChange llProperty llValue
End_Procedure
```

Visual Objects
```
METHOD OCX_EditChange(Property,Value) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_EditChange(COM _Property,COMVariant /*variant*/ _Value)
{
}
```

XBasic
```
function EditChange as v (Property as
OLE::Exontrol.PropertiesList.1::IProperty,Value as A)
end function
```

dBASE
```
function nativeObject_EditChange(Property,Value)
return
```

The following sample prints the current text of the control's editing box:

```
Private Sub PropertiesList1_EditChange(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Value As Variant)
    Debug.Print Value
End Sub
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

Type	Description
EventID as Long	A Long expression that specifies the identifier of the event. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties).

The Event notification occurs ANY time the control fires an event.

This is useful for X++ language, which does not support event with parameters passed by reference.

In X++ the "Error executing code: FormActiveXControl (data source), method ... called with invalid parameters" occurs when handling events that have parameters passed by reference. Passed by reference, means that in the event handler, you can change the value for that parameter, and so the control will takes the new value, and use it. The X++ is NOT able to handle properly events with parameters by reference, so we have the solution.

The solution is using and handling the Event notification and EventParam method., instead handling the event that gives the "invalid parameters" error executing code.

Let's presume that we need to handle the BarParentChange event to change the _Cancel parameter from false to true, which fires the "Error executing code: FormActiveXControl (data source), method onEvent_BarParentChange called with invalid parameters." We need to know the identifier of the BarParentChange event (each event has an unique identifier and it is static, defined in the control's type library). If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print expropertieslist1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
```

```
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

Now, we can see that the identifier for the BarParentChange event is 125, so we need to handle the Event event as:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
        expropertieslist1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

The code checks if the BarParentChange (_EventID == 125) event is fired, and changes the third parameter of the event to true. The definition for BarParentChange event can be consulted in the control's documentation or in the ActiveX explorer. So, anytime you need to access the original parameters for the event you should use the EventParam method that allows you to get or set a parameter. If the parameter is not passed by reference, you can not change the parameter's value.

Now, let's add some code to see a complex sample, so let's say that we need to prevent moving the bar from an item to any disabled item. So, we need to specify the Cancel parameter as not Items.EnableItem(NewItem), in other words cancels if the new parent is disabled. Shortly the code will be:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
        if ( !expropertieslist1.Items().EnableItem( expropertieslist1.EventParam( 2 /*NewItem*/ ) ) )
            expropertieslist1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
```

```
}
```

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

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,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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 IncludeProperty (Property as Property, Cancel as Boolean)

Fired when the properties browser is about to include a new property.

Type	Description
Property as Property	A Property object that contains information about the item that is going to be included.
Cancel as Boolean	A boolean expression that indicates whether the property is excluded or included.

Use the IncludeProperty event to filter the object properties. The event is fired only if the FireIncludeProperty property is True. Use the Cancel argument of the event to include or exclude a property. Use Cancel = True to exclude a property from the list. Use the HTMLName property to display HTML format in the Name column.

Syntax for IncludeProperty event, **/NET** version, on:

C#
```
private void IncludeProperty(object
sender,exontrol.EXPROPERTIESLISTLib.Property Property,ref bool Cancel)
{
}
```

VB
```
Private Sub IncludeProperty(ByVal sender As System.Object,ByVal Property As
exontrol.EXPROPERTIESLISTLib.Property,ByRef Cancel As Boolean) Handles
IncludeProperty
End Sub
```

Syntax for IncludeProperty event, **/COM** version, on:

C#
```
private void IncludeProperty(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_IncludePropertyEvent e)
{
}
```

C++
```
void OnIncludeProperty(LPDISPATCH Property,BOOL FAR* Cancel)
{
}
```

C++ Builder
```
void __fastcall IncludeProperty(TObject *Sender,Expropertieslistlib_tlb::IProperty
*Property,VARIANT_BOOL * Cancel)
```

```
{
}
```

Delphi

```
procedure IncludeProperty(ASender: TObject; Property : IProperty;var Cancel :
WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure IncludeProperty(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_IncludePropertyEvent);
begin
end;
```

Powe…

```
begin event IncludeProperty(oleobject Property,boolean Cancel)
end event IncludeProperty
```

VB.NET

```
Private Sub IncludeProperty(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_IncludePropertyEvent) Handles
IncludeProperty
End Sub
```

VB6

```
Private Sub IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty,Cancel As Boolean)
End Sub
```

VBA

```
Private Sub IncludeProperty(ByVal Property As Object,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Property,Cancel

```

Xbas…

```
PROCEDURE OnIncludeProperty(oPropertiesList,Property,Cancel)
RETURN
```

Syntax for IncludeProperty event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="IncludeProperty(Property,Cancel)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function IncludeProperty(Property,Cancel)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComIncludeProperty Variant llProperty Boolean llCancel
    Forward Send OnComIncludeProperty llProperty llCancel
End_Procedure
```

Visual Objects
```
METHOD OCX_IncludeProperty(Property,Cancel) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_IncludeProperty(COM _Property,COMVariant /*bool*/ _Cancel)
{
}
```

XBasic
```
function IncludeProperty as v (Property as
OLE::Exontrol.PropertiesList.1::IProperty,Cancel as L)
end function
```

dBASE
```
function nativeObject_IncludeProperty(Property,Cancel)
return
```

For instance, if the control's ShowVariables is True, the control includes also the variables for an IPictureDisp object (hPal variable).

The following sample shows how to exclude variables of IPictureDisp (picture) properties:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
    If Not (Property.Object Is Nothing) Then
        Cancel = Property.Variable And TypeOf Property.Object Is IPictureDisp
    End If
End Sub
```

The following sample includes only the properties of IFontDisp type, and their variables:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
```

```
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not Property.Type = "Font*"
   If (Cancel) Then
      If Not (Property.Object Is Nothing) Then
         Cancel = Not Property.Variable
      End If
   End If
End Sub
```

The following sample include only the properties of boolean type, and properties of Object type:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not Property.Type = "BOOL"
   If (Cancel) Then
      Cancel = Not Property.PropertyObject
   End If
End Sub
```

The following sample includes all properties contained by "Misc" category:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not (Property.CategoryName = "Misc")
End Sub
```

The following sample shows how to simulate the VB browser (the ShowObjects property is True):

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Property.PropertyObject
   If (Cancel) Then
      Cancel = Not (Property.Type = "Font*" Or Property.Type = "Picture*")
   End If
End Sub
```

The following sample shows how to include into your browser only the property pages:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not Property.PropertyPage
End Sub
```

Here's a sample that shows how to include only hidden members:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   Cancel = Not (Property.Flags And &H40) = &H40
End Sub
```

The following sample excludes the "hPal" variable of a Picture property:

```
Private Sub PropertiesList1_IncludeProperty(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Cancel As Boolean)
   If Property.Variable = True Then
      If Property.Name = "hPal" Then
         Cancel = True
      End If
   End If
End Sub
```

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 stemporary integer variable and then comparing shift to a bit mask. The control fires the EditChange event while user types characters in the property's text box control. 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,
```

```
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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, Shift 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.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the key specified in the KeyAsci argument is pressed or released.

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress does not indicate the physical state of the keyboard; instead, it passes a character. KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters

Syntax for KeyPress event, **/NET** version, on:

C#

```
private void KeyPress(object sender,ref short KeyAscii,short Shift)
{
}
```

VB

```
Private Sub KeyPress(ByVal sender As System.Object,ByRef KeyAscii As Short,ByVal
Shift As Short) Handles KeyPress
End Sub
```

Syntax for KeyPress event, **/COM** version, on:

C#

```
private void KeyPressEvent(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_KeyPressEvent e)
{
}
```

C++

```
void OnKeyPress(short FAR* KeyAscii,short Shift)
{
}
```

C++ Builder

```
void __fastcall KeyPress(TObject *Sender,short * KeyAscii,short Shift)
{
}
```

Delphi

```
procedure KeyPress(ASender: TObject; var KeyAscii : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyPressEvent(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_KeyPressEvent);
begin
end;
```

Powe…

```
begin event KeyPress(integer KeyAscii,integer Shift)
end event KeyPress
```

VB.NET

```
Private Sub KeyPressEvent(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_KeyPressEvent) Handles
KeyPressEvent
End Sub
```

VB6

```
Private Sub KeyPress(KeyAscii As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyPress(KeyAscii As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyAscii,Shift

```

Xbas…

```
PROCEDURE OnKeyPress(oPropertiesList,KeyAscii,Shift)
RETURN
```

Syntax for KeyPress event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="KeyPress(KeyAscii,Shift)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyPress(KeyAscii,Shift)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyPress Short llKeyAscii Short llShift
    Forward Send OnComKeyPress llKeyAscii llShift
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyPress(KeyAscii,Shift) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii,int _Shift)
{
}
```

XBasic

```
function KeyPress as v (KeyAscii as N,Shift as N)
end function
```

dBASE

```
function nativeObject_KeyPress(KeyAscii,Shift)
return
```

Use the KeyPress event to handle keyboard events. The following sample shows how to handle Delete key:

```
Private Sub PropertiesList1_KeyPress(KeyAscii As Integer, Shift As Integer)
    If KeyAscii = KeyCodeConstants.vbKeyDelete Then
        MsgBox PropertiesList1.SelectedProperty.Name
    End If
End Sub
```

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. The control fires the KeyDown event when user presses a key.

Syntax for KeyUp event, **/NET** version, on:

C#

```
private void KeyUp(object sender,ref short KeyCode,short Shift)
{
}
```

VB

```
Private Sub KeyUp(ByVal sender As System.Object,ByRef KeyCode As Short,ByVal
Shift As Short) Handles KeyUp
End Sub
```

Syntax for KeyUp event, **/COM** version, on:

C#

```
private void KeyUpEvent(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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 ModalPropertyChange (Property as Property, Value as Variant, Cancel as Boolean)

Fired when the properties browser is about to change a property's value using a modal dialog.

Type	Description
Property as Property	A Property object being changed using a modal dialog.
Value as Variant	A Variant expression that indicates the newly property's value.
Cancel as Boolean	A boolean expression that indicates whether the control disables or enables the default implementation.

Use the ModalPropertyChange event to replace the default implementation of modal type editors (IFontDisp (font) properties, IPictureDisp (picture) properties, object properties pages, EditPage, EditColorPage, EditButton types). The "Invalid property value" message is displayed if the Property does not accept the Value. To avoid showing error messages set the InvalidValueMessage property to an empty string.

Syntax for ModalPropertyChange event, **/NET** version, on:

C#
```
private void ModalPropertyChange(object
sender,exontrol.EXPROPERTIESLISTLib.Property Property,ref object Value,ref bool
Cancel)
{
}
```

VB
```
Private Sub ModalPropertyChange(ByVal sender As System.Object,ByVal Property
As exontrol.EXPROPERTIESLISTLib.Property,ByRef Value As Object,ByRef Cancel As
Boolean) Handles ModalPropertyChange
End Sub
```

Syntax for ModalPropertyChange event, **/COM** version, on:

C#
```
private void ModalPropertyChange(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ModalPropertyChangeEvent e)
{
}
```

C++
```
void OnModalPropertyChange(LPDISPATCH Property,VARIANT FAR* Value,BOOL
```

```
FAR* Cancel)
{
}
```

C++ Builder

```
void __fastcall ModalPropertyChange(TObject
*Sender,Expropertieslistlib_tlb::IProperty *Property,Variant * Value,VARIANT_BOOL
* Cancel)
{
}
```

Delphi

```
procedure ModalPropertyChange(ASender: TObject; Property : IProperty;var
Value : OleVariant;var Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ModalPropertyChange(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ModalPropertyChangeEvent);
begin
end;
```

Powe…

```
begin event ModalPropertyChange(oleobject Property,any Value,boolean Cancel)
end event ModalPropertyChange
```

VB.NET

```
Private Sub ModalPropertyChange(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ModalPropertyChangeEvent)
Handles ModalPropertyChange
End Sub
```

VB6

```
Private Sub ModalPropertyChange(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty,Value As Variant,Cancel As Boolean)
End Sub
```

VBA

```
Private Sub ModalPropertyChange(ByVal Property As Object,Value As
Variant,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Property,Value,Cancel
```

Xbas…
```
PROCEDURE OnModalPropertyChange(oPropertiesList,Property,Value,Cancel)
RETURN
```

Syntax for ModalPropertyChange event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="ModalPropertyChange(Property,Value,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ModalPropertyChange(Property,Value,Cancel)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComModalPropertyChange Variant llProperty Variant llValue
Boolean llCancel
    Forward Send OnComModalPropertyChange llProperty llValue llCancel
End_Procedure
```

Visual Objects
```
METHOD OCX_ModalPropertyChange(Property,Value,Cancel) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ModalPropertyChange(COM _Property,COMVariant /*variant*/
_Value,COMVariant /*bool*/ _Cancel)
{
}
```

XBasic
```
function ModalPropertyChange as v (Property as
OLE::Exontrol.PropertiesList.1::IProperty,Value as A,Cancel as L)
end function
```

dBASE
```
function nativeObject_ModalPropertyChange(Property,Value,Cancel)
return
```

The following sample replaces the editor for properties of IPictureDisp (picture) type (the sample uses Type property of the Property to check the property's type. The sample doesn't use the statement typeof Property.Object is IPictureDisp because the Object

property might be set to nothing, and so the operator typeof will be unable to determine the type of the object):

```
Private Sub PropertiesList1_ModalPropertyChange(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Value As Variant, Cancel As Boolean)
    If Property.Type = "Picture*" Then
        MsgBox "Invoke your dialog here, and change the Value parameter, when your dialog
is closed."
        Value = StdFunctions.LoadPicture("c:\winnt\system32\setup.bmp")
        Cancel = True
    End If
End Sub
```

The following sample shows how to change the default font editor:

```
Private Sub PropertiesList1_ModalPropertyChange(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Value As Variant, Cancel As Boolean)
    If (Property.Type = "Font*") Then
        MsgBox "Use your implementation here"
        Set Value = Me.Font
        Cancel = True
    End If
End Sub
```

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.

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,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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.PropertiesList.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.PropertiesList.1::OLE_YPOS_PIXELS)
end function
```

dBASE
```
function nativeObject_MouseDown(Button,Shift,X,Y)
return
```

event MouseMove (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user moves the mouse.

Type	Description
Button as Integer	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders.

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,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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.PropertiesList.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.PropertiesList.1::OLE_YPOS_PIXELS)
end function
```

dBASE
```
function nativeObject_MouseMove(Button,Shift,X,Y)
return
```

event MouseUp (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user releases a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

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,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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(oPropertiesList,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.PropertiesList.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.PropertiesList.1::OLE_YPOS_PIXELS)
end function
```

dBASE
```
function nativeObject_MouseUp(Button,Shift,X,Y)
return
```

event PropertyChange (Property as Property, Value as Variant)

Fired when the properties browser is about to change the property's value.

Type	Description
Property as Property	A Property object that was changed using one of the control's editors, and it is about to change the property's value.
Value as Variant	A Variant expression that specifies the newly value that will be passed to the property.

The PropertyChange event notifies your application that user is about to change a property. If you are going to change the Value parameter during this event, make sure that your value is accepted by the property, else the control will popup an error message: "Invalid property value". To avoid showing error messages by the control when changing properties, set the InvalidValueMessage property to empty message. The PropertyChange event is called, even if the user canceled the default implementation in ModalPropertyChange event. The control fires the EditChange event when user types characters in the property's text box control. The PropertyChanged event is fired after changing the value of the property. Use the Value property to retrieve the property's value. Use the Name property to retrieve the name of the property. Use the ID property to identify a property by its identifier.

Syntax for PropertyChange event, **/NET** version, on:

C#
```
private void PropertyChange(object
sender,exontrol.EXPROPERTIESLISTLib.Property Property,ref object Value)
{
}
```

VB
```
Private Sub PropertyChange(ByVal sender As System.Object,ByVal Property As
exontrol.EXPROPERTIESLISTLib.Property,ByRef Value As Object) Handles
PropertyChange
End Sub
```

Syntax for PropertyChange event, **/COM** version, on:

C#
```
private void PropertyChange(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_PropertyChangeEvent e)
{
}
```

C++

```
void OnPropertyChange(LPDISPATCH Property,VARIANT FAR* Value)
{
}
```

C++ Builder

```
void __fastcall PropertyChange(TObject *Sender,Expropertieslistlib_tlb::IProperty
*Property,Variant * Value)
{
}
```

Delphi

```
procedure PropertyChange(ASender: TObject; Property : IProperty;var Value :
OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure PropertyChange(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_PropertyChangeEvent);
begin
end;
```

Powe…

```
begin event PropertyChange(oleobject Property,any Value)
end event PropertyChange
```

VB.NET

```
Private Sub PropertyChange(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_PropertyChangeEvent) Handles
PropertyChange
End Sub
```

VB6

```
Private Sub PropertyChange(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty,Value As Variant)
End Sub
```

VBA

```
Private Sub PropertyChange(ByVal Property As Object,Value As Variant)
End Sub
```

VFP

```
LPARAMETERS Property,Value

```

Xbas…

```
PROCEDURE OnPropertyChange(oPropertiesList,Property,Value)
```

```
RETURN
```

Syntax for PropertyChange event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="PropertyChange(Property,Value)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function PropertyChange(Property,Value)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComPropertyChange Variant llProperty Variant llValue
    Forward Send OnComPropertyChange llProperty llValue
End_Procedure
```

Visual Objects
```
METHOD OCX_PropertyChange(Property,Value) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_PropertyChange(COM _Property,COMVariant /*variant*/ _Value)
{
}
```

XBasic
```
function PropertyChange as v (Property as
OLE::Exontrol.PropertiesList.1::IProperty,Value as A)
end function
```

dBASE
```
function nativeObject_PropertyChange(Property,Value)
return
```

The following sample prints the property's name and the property's newly value when user changes the property:

```
Private Sub PropertiesList1_PropertyChange(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty, Value As Variant)
    Debug.Print "The """ & Property.Name & """'s value is changed. The newly value is " &
Value

```

```
End Sub
```

event PropertyChanged (Property as Property)

Occurs after the property's value is changed.

Type	Description
Property as Property	A Property object being changed

The PropertyChanged event notifies your application that the user changes the value of the property. Use the Value property to retrieve the property's value. Use the Name property to retrieve the name of the property. Use the ID property to identify a property by its identifier. The PropertyChange event is fired before changing the value of the property.

Syntax for PropertyChanged event, **/NET** version, on:

C#
```
private void PropertyChanged(object
sender,exontrol.EXPROPERTIESLISTLib.Property Property)
{
}
```

VB
```
Private Sub PropertyChanged(ByVal sender As System.Object,ByVal Property As
exontrol.EXPROPERTIESLISTLib.Property) Handles PropertyChanged
End Sub
```

Syntax for PropertyChanged event, **/COM** version, on:

C#
```
private void PropertyChanged(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_PropertyChangedEvent e)
{
}
```

C++
```
void OnPropertyChanged(LPDISPATCH Property)
{
}
```

C++ Builder
```
void __fastcall PropertyChanged(TObject *Sender,Expropertieslistlib_tlb::IProperty
*Property)
{
}
```

Delphi
```
procedure PropertyChanged(ASender: TObject; Property : IProperty);
```

```
begin
end;
```

Delphi 8 (.NET only)

```
procedure PropertyChanged(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_PropertyChangedEvent);
begin
end;
```

Powe…

```
begin event PropertyChanged(oleobject Property)
end event PropertyChanged
```

VB.NET

```
Private Sub PropertyChanged(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_PropertyChangedEvent) Handles
PropertyChanged
End Sub
```

VB6

```
Private Sub PropertyChanged(ByVal Property As
EXPROPERTIESLISTLibCtl.IProperty)
End Sub
```

VBA

```
Private Sub PropertyChanged(ByVal Property As Object)
End Sub
```

VFP

```
LPARAMETERS Property

```

Xbas…

```
PROCEDURE OnPropertyChanged(oPropertiesList,Property)
RETURN
```

Syntax for PropertyChanged event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="PropertyChanged(Property)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function PropertyChanged(Property)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComPropertyChanged Variant llProperty
    Forward Send OnComPropertyChanged llProperty
End_Procedure
```

Visual Objects

```
METHOD OCX_PropertyChanged(Property) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_PropertyChanged(COM _Property)
{
}
```

XBasic

```
function PropertyChanged as v (Property as
OLE::Exontrol.PropertiesList.1::IProperty)
end function
```

dBASE

```
function nativeObject_PropertyChanged(Property)
return
```

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.EXPROPERTIESLISTLib.ScrollBarEnum
ScrollBar,exontrol.EXPROPERTIESLISTLib.ScrollPartEnum ScrollPart)
{
}
```

VB
```
Private Sub ScrollButtonClick(ByVal sender As System.Object,ByVal ScrollBar As
exontrol.EXPROPERTIESLISTLib.ScrollBarEnum,ByVal ScrollPart As
exontrol.EXPROPERTIESLISTLib.ScrollPartEnum) Handles ScrollButtonClick
End Sub
```

Syntax for ScrollButtonClick event, **/COM** version, on:

C#
```
private void ScrollButtonClick(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ScrollButtonClickEvent e)
{
}
```

C++
```
void OnScrollButtonClick(long ScrollBar,long ScrollPart)
{
```

```
}
```

C++ Builder

```
void __fastcall ScrollButtonClick(TObject
*Sender,Expropertieslistlib_tlb::ScrollBarEnum
ScrollBar,Expropertieslistlib_tlb::ScrollPartEnum ScrollPart)
{
}
```

Delphi

```
procedure ScrollButtonClick(ASender: TObject; ScrollBar :
ScrollBarEnum;ScrollPart : ScrollPartEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ScrollButtonClick(sender: System.Object; e:
AxEXPROPERTIESLISTLib._IPropertiesListEvents_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
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ScrollButtonClickEvent) Handles
ScrollButtonClick
End Sub
```

VB6

```
Private Sub ScrollButtonClick(ByVal ScrollBar As
EXPROPERTIESLISTLibCtl.ScrollBarEnum,ByVal ScrollPart As
EXPROPERTIESLISTLibCtl.ScrollPartEnum)
End Sub
```

VBA

```
Private Sub ScrollButtonClick(ByVal ScrollBar As Long,ByVal ScrollPart As Long)
End Sub
```

VFP

```
LPARAMETERS ScrollBar,ScrollPart

```

Xbas…

```
PROCEDURE OnScrollButtonClick(oPropertiesList,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.PropertiesList.1::ScrollBarEnum,ScrollPart as
OLE::Exontrol.PropertiesList.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 PropertiesList1
   .BeginUpdate
      .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
```

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

```
Private Sub PropertiesList1_ScrollButtonClick(ByVal ScrollPart As
EXPROPERTIESLISTLibCtl.ScrollPartEnum)
  MsgBox (ScrollPart)
End Sub
```

The following VB.NET sample displays the identifier of the scroll's button being clicked:

```
With AxPropertiesList1
  .BeginUpdate()
  .set_ScrollPartVisible(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part Or
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, True)
  .set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
  .set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
  .EndUpdate()
End With
```

```
Private Sub AxPropertiesList1_ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ScrollButtonClickEvent) Handles
AxPropertiesList1.ScrollButtonClick
    MessageBox.Show( e.scrollPart.ToString())
End Sub
```

The following C# sample displays the identifier of the scroll's button being clicked:

```
axPropertiesList1.BeginUpdate();
axPropertiesList1.set_ScrollPartVisible(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part |
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, true);
axPropertiesList1.set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
```

```
axPropertiesList1.set_ScrollPartCaption(EXPROPERTIESLISTLib.ScrollBarEnum.exVScroll,
EXPROPERTIESLISTLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axPropertiesList1.EndUpdate();
```

```
private void axPropertiesList1_ScrollButtonClick(object sender,
AxEXPROPERTIESLISTLib._IPropertiesListEvents_ScrollButtonClickEvent e)
{
   MessageBox.Show(e.scrollPart.ToString());
}
```

The following C++ sample displays the identifier of the scroll's button being clicked:

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

```
void OnScrollButtonClickPropertiesList1(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.PropertiesList1
   .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))
```

event SelChange ()

Fired when the selected property is changed.

Type	Description

The SelChange event notifies your application that user changes the selection. The SelChange event is not called when the control browses a new object using the Select method. The SelectedProperty property gets the selected property. The Name property gets the property's name. The Value property gets the property's value.

Syntax for SelChange event, **/NET** version, on:

C#

```
private void SelChange(object sender)
{
}
```

VB

```
Private Sub SelChange(ByVal sender As System.Object) Handles SelChange
End Sub
```

Syntax for SelChange event, **/COM** version, on:

C#

```
private void SelChange(object sender, EventArgs e)
{
}
```

C++

```
void OnSelChange()
{
}
```

C++ Builder

```
void __fastcall SelChange(TObject *Sender)
{
}
```

Delphi

```
procedure SelChange(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure SelChange(sender: System.Object; e: System.EventArgs);
begin
```

```
end;
```

Powe…
```
begin event SelChange()
end event SelChange
```

VB.NET
```
Private Sub SelChange(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SelChange
End Sub
```

VB6
```
Private Sub SelChange()
End Sub
```

VBA
```
Private Sub SelChange()
End Sub
```

VFP
```
LPARAMETERS nop

```

Xbas…
```
PROCEDURE OnSelChange(oPropertiesList)
RETURN
```

Syntax for SelChange event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="SelChange()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function SelChange()
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComSelChange
    Forward Send OnComSelChange
End_Procedure
```

Visual Objects
```
METHOD OCX_SelChange() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_SelChange()
{
}
```

XBasic

```
function SelChange as v ()
end function
```

dBASE

```
function nativeObject_SelChange()
return
```

The following sample prints the name and the type of the selected property (for instance, the sample is useful to find out the type of the property selected, when you need to include or exclude properties using the IncludeProperty event):

```
Private Sub PropertiesList1_SelChange()
    Debug.Print "You have selected the """ & PropertiesList1.SelectedProperty.Name & """.
The type for it is: " & PropertiesList1.SelectedProperty.Type
End Sub
```