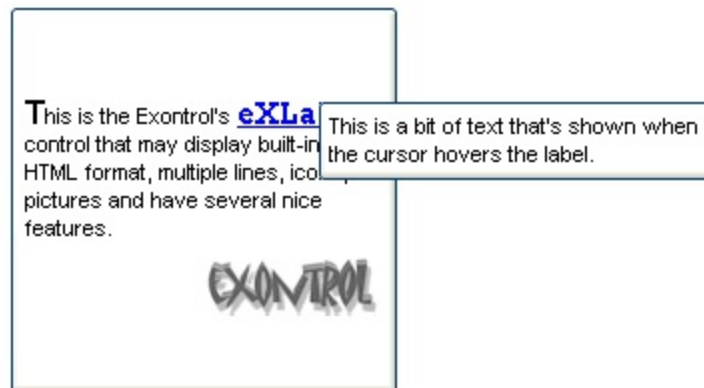# ExLabel

The Exontrol's eXLabel component provides HTML labels for your forms or dialogs. A label control displays static text on a forms and allows you to manipulate it programmatically. The eXLabel control easily replaces the Standard Windows label by supporting most of the same properties, methods and events. In addition, you have complete control over how the label is to be displayed.

Features include:

- WYSWYG Template/Layout Editor support.
- Skinnable Interface support (ability to apply a skin to any background part)
- Built-In Multiple-Lines HTML support.
- Horizontal, Vertical, Rotate, Mirror support.
- Multiple-Lines HTML Tooltip support.
- Unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the label's background.
- Text Decorations support, like gradient, outlined characters, shadow, and so on
- Ability to use the label with a Transparent background
- Ability to display and handle anchor/link elements
- Ability to assign icons, pictures anywhere in the caption.
- Ability to specify a picture on the label's background

How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

The Column object uses the AlignmentEnum enumeration to align the caption in the label. The [Alignment](#) property aligns the caption in the label.

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. The [Appearance](#) property changes the control's border.

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 BackgroundExtPropertyEnum

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

Name	Value	Description
exToStringExt	0	Specifies the part's ToString representation. The BackgroundExt property specifies the EBN String format to be displayed on the object's background. *The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the To String field.* *Sample:* *"client(right[18] (bottom[18,pattern=6,frame=0,framethick]),bottom[4 (bottom[18,pattern=6,frame=0,framethick])"* *generates the following layout:* *where it is applied to an object it looks as follows:* *(String expression, read-only).*
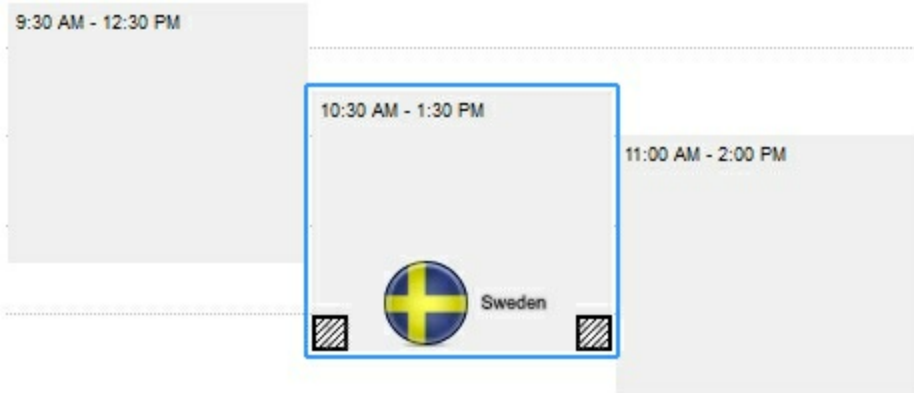
exBackColorExt	1	Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000.* *(Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*

Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect.*

Based on the exAnchorExt value the exClientExt is:

- *0 (**none**, the object is not anchored to any side), the format of the exClientExt is **"left,top,width,height"** (as string) where (left,top) margin indicates the position where the part starts, and the (width,height) pair specifies its size. The left, top, width or height could be any expression (+,-,/ or *) that can include numbers associated with pixels or percents. For instance: "25%,25%,50%,50%" indicates the middle of the parent object, and so when the parent is resized the client is resized accordingly. The "50%-8,50%-8,16,16" value specifies that the size of the object is always 16x16 pixels and positioned on the center of the parent object.*
- *1 (**left**, the object is anchored to left side of the parent), the format of the exClientExt is **width** (string or numeric) where width indicates the width of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates*

exClientExt	2	*the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.*

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

Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent.

(String/Numeric expression)

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

FormatAnchor property customizes the visual effect for anchor elements.

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as " <a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAA " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAC string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

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

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present,

the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.

- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the

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

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.

- **&** glyph characters as **&amp;** (&), **&lt;** (<), **&gt;** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the &#8364; 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 &lt;b&gt;bold&lt;/b&gt;

- **<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:

(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.*
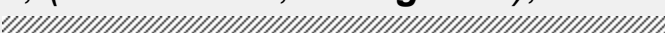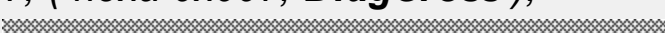
- *32, (hexa 0x20, **Bottom-Left**), Text is vertically aligned at the bottom, and horizontally aligned on the left.*
- *33, (hexa 0x21, **Bottom-Center**), Text is vertically aligned at the bottom, and horizontally aligned at the center.*
- *34, (hexa 0x22, **Bottom-Right**), Text is vertically aligned 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**),*

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.

Name	Value	Description
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 [ToolTipText](#) property to specify the cell's tooltip. Use the [ToolTipWidth](#) property to specify the width of the tooltip window. The [ToolTipDelay](#) property specifies the time in ms that passes before the ToolTip appears. Use the [ShowToolTip](#) method to display a custom tooltip.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.

constants HTMLRotateEnum

The HTMLRotateEnum expression specifies how the control displays the HTML caption. The [Rotate](#) property rotates the HTML caption. The HTMLRotateEnum expression supports the following values:

Name	Value	Description
exHTMLHorizontal	0	Specifies that the HTML text is horizontally displayed. This flag can be combined with exHTMLMirror flag.
exHTMLVertical	1	Specifies that the HTML text is vertically displayed. This flag can be combined with exHTMLMirror flag.
exHTMLMirror	16	Specifies that the HTML text is displayed in mirror. This flag can be combined with exHTMLHorizontal or exHTMLVertical flag.

constants IndexExtEnum

The IndexExtEnum type specifies the index of the part of the EBN object to be accessed. The Index parameter of the BackgroundExtValue property indicates the index of the part of the EBN object to be changed or accessed. *The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the To String field. The list of objects that compose the EBN are displayed on the left side of the Builder tool, and the Index of the part is displayed on each item aligned to the right as shown in the following screen shot:*
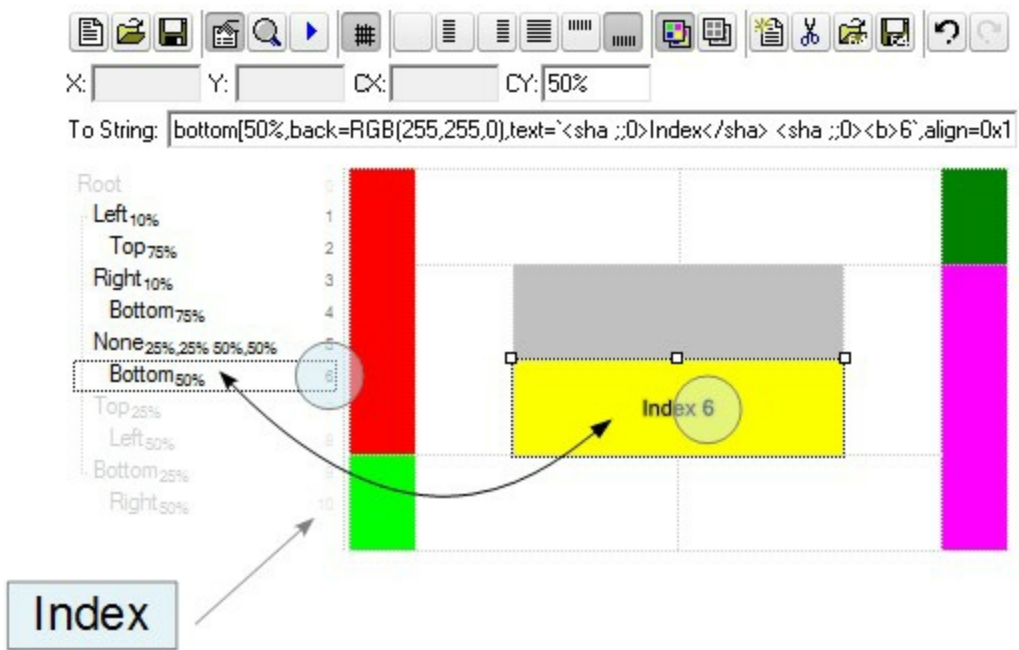
In this sample, there are 11 objects that composes the EBN, so the Index property goes from 0 which indicates the root, and 10, which is the last item in the list

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

Before calling the BackgroundExt property:
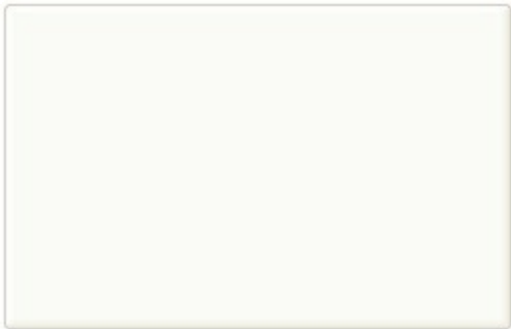
After calling the BackgroundExt property:

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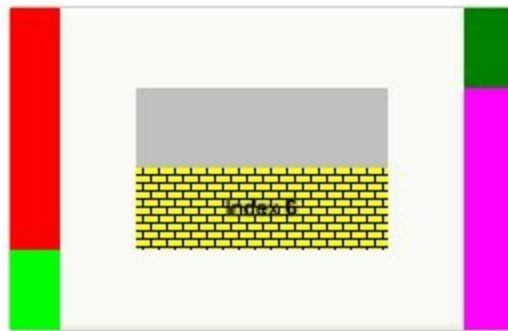
The IndexExtEnum type supports the following values:

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

constants PictureDisplayEnum

Specifies how the picture is displayed on the control's background. Use the [PictureDisplay](#) property to specify how the control displays its picture.

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

constants VAlignmentEnum

The VAlignmentEnum type specifies the vertical alignment of the caption inside the label. Use the [VAlignment](#) property to specify the vertical alignment of the caption inside the label.

Name	Value	Description
TopAlignment	0	Top Alignment
MiddleAlignment	1	Middle Alignment
BottomAlignment	2	Bottom Alignment

Appearance object

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

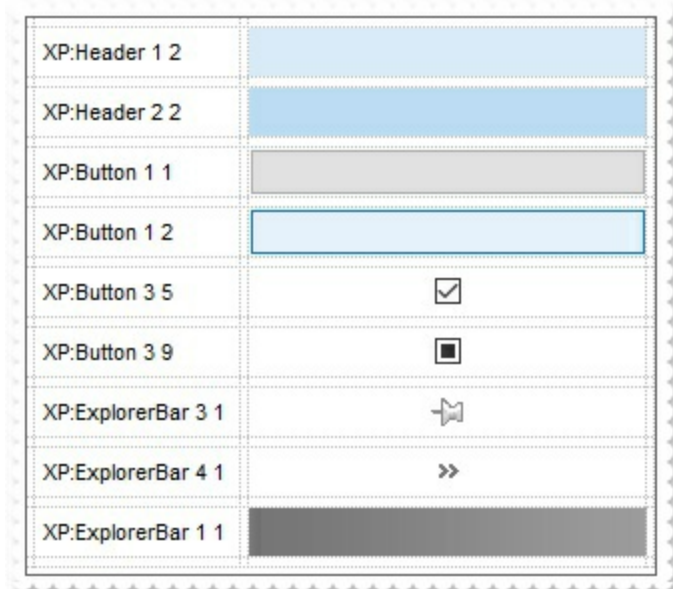| Name	Description
Add	Adds or replaces a skin object to the control.
Clear	Removes all skins in the control.
Remove	Removes a specific skin from the control.

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

Adds or replaces a skin object to the control

Type	Description
ID as Long	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

XP:Header 1 2	
XP:Header 2 2	
XP:Button 1 1	
XP:Button 1 2	
XP:Button 3 5	☑
XP:Button 3 9	◼
XP:ExplorerBar 3 1	
XP:ExplorerBar 4 1	»
XP:ExplorerBar 1 1	

XP:Header 1 2	
XP:Header 2 2	
XP:Button 1 1	
XP:Button 1 2	
XP:Button 3 5	☑
XP:Button 3 9	◼
XP:ExplorerBar 3 1	
XP:ExplorerBar 4 1	»
XP:ExplorerBar 1 1	
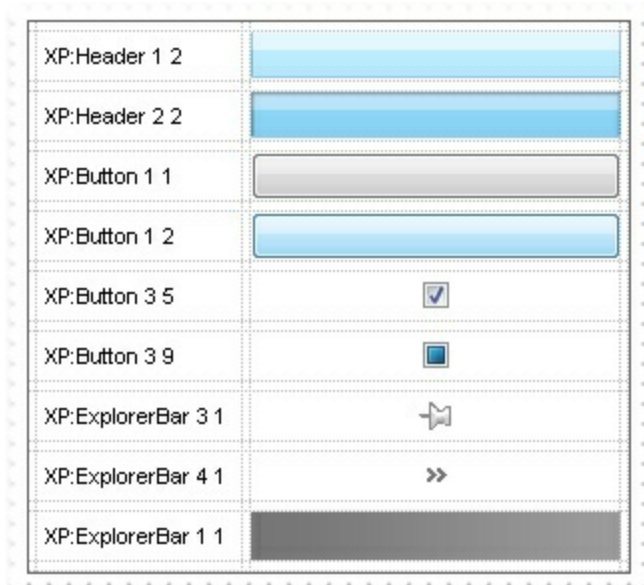
- A copy of another skin with different coordinates (position, size), if the Skin parameter starts with "**CP**:". Use this option, to display the EBN, using different coordinates (position, size). By default, the EBN skin object is rendered on the part's client area. Using this option, you can display 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:
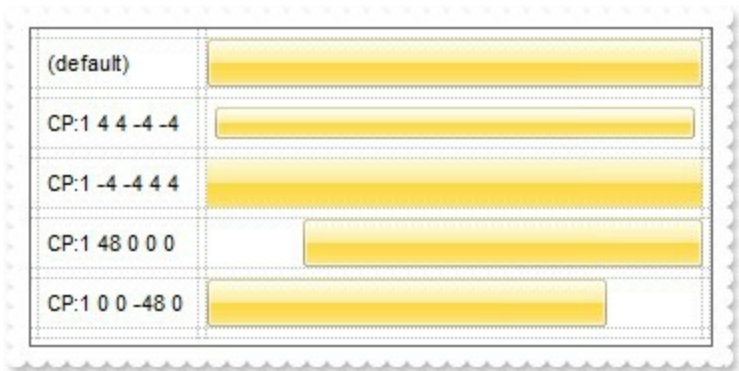
Return	Description
Boolean	A Boolean expression that indicates whether the new skin was added or replaced.

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.

This is the Exontrol's **eXLa[bel]** control that may display built-in HTML format, multiple lines, icons, pictures and have several nice features.

This is a bit of text that's shown when the cursor hovers the label.

EXONTROL

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 Appearance = Appearance Or &H2000000 indicates that we apply the skin with the index 2 using the old color, to the label's border.

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

- control's **borders** using the Appearance property
- **tooltips**, Background property
- Any HTML caption that includes an <img> tag.

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

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

		CBS_MIXEDPRESS CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
	BP_PUSHBUTTON = 1	PBS_NORMAL = 1 = 2 PBS_PRESSEI PBS_DISABLED = PBS_DEFAULTED :
	BP_RADIOBUTTON = 2	RBS_UNCHECKED 1 RBS_UNCHECKE RBS_UNCHECKED = 3 RBS_UNCHECKED = 4 RBS_CHECKEI 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 = 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_SELECTEI EBHP_SELECTEDF 6

	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSEI
	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 2 HIS_PRESSED =
	HP_HEADERITEMLEFT = 2	HILS_NORMAL = 1 = 2 HILS_PRESSEI
	HP_HEADERITEMRIGHT = 3	HIRS_NORMAL = 1 = 2 HIRS_PRESSE
	HP_HEADERSORTARROW = 4	HSAS_SORTEDUP HSAS_SORTEDDC
LISTVIEW	LVP_EMPTYTEXT = 5	
	LVP_LISTDETAIL = 3	
	LVP_LISTGROUP = 2	
	LVP_LISTITEM = 1	LIS_NORMAL = 1 L 2 LIS_SELECTED = LIS_DISABLED = 4 LIS_SELECTEDNO 5
	LVP_LISTSORTEDDETAIL = 4	
MENU	MP_MENUBARDROPDOWN = 4	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3 MS_NORMAL = 1

	MP_MENUBARITEM = 3	MS_SELECTED = 2
		MS_DEMOTED = 3
		MS_NORMAL = 1
	MP_CHEVRON = 5	MS_SELECTED = 2
		MS_DEMOTED = 3
		MS_NORMAL = 1
	MP_MENUDROPDOWN = 2	MS_SELECTED = 2
		MS_DEMOTED = 3
		MS_NORMAL = 1
	MP_MENUITEM = 1	MS_SELECTED = 2
		MS_DEMOTED = 3
		MS_NORMAL = 1
	MP_SEPARATOR = 6	MS_SELECTED = 2
		MS_DEMOTED = 3
		MDS_NORMAL = 1
		= 2 MDS_PRESSED
MENUBAND	MDP_NEWAPPBUTTON = 1	MDS_DISABLED =
		MDS_CHECKED =
		MDS_HOTCHECKE
	MDP_SEPERATOR = 2	
		DNS_NORMAL = 1
PAGE	PGRP_DOWN = 2	= 2 DNS_PRESSED
		DNS_DISABLED =
		DNHZS_NORMAL =
		DNHZS_HOT = 2
	PGRP_DOWNHORZ = 4	DNHZS_PRESSED
		DNHZS_DISABLED
		UPS_NORMAL = 1
	PGRP_UP = 1	= 2 UPS_PRESSED
		UPS_DISABLED =
		UPHZS_NORMAL =
		UPHZS_HOT = 2
	PGRP_UPHORZ = 3	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

RP_CHEVRONVERT = 5

RP_GRIPPER = 1
RP_GRIPPERVERT = 2

CHEVS_HOT = 2
CHEVS_PRESSED

ABS_DOWNDISAB
ABS_DOWNHOT,
ABS_DOWNNORM
ABS_DOWNPRES
ABS_UPDISABLED
ABS_UPHOT,
ABS_UPNORMAL,
ABS_UPPRESSED,
ABS_LEFTDISABLI
ABS_LEFTHOT,
ABS_LEFTNORMA
ABS_LEFTPRESSE
ABS_RIGHTDISAB
ABS_RIGHTHOT,
ABS_RIGHTNORM
ABS_RIGHTPRES

SCROLLBAR   SBP_ARROWBTN = 1

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

	SBP_UPPERTRACKHORZ = 5	SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
		SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKVERT = 7	
	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	

TAB	SP_GRIPPERPANE = 2	
	TABP_BODY = 10	
	TABP_PANE = 9	
	TABP_TABITEM = 1	TIS_NORMAL = 1 TIS_HOT = 2 TIS_SELECTED = 3 TIS_DISABLED = 4 TIS_FOCUSED = 5
	TABP_TABITEMBOTHEDGE = 4	TIBES_NORMAL = 1 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 =
	TTP_STANDARD = 1	TTSS_NORMAL = 1 TTSS_LINK = 2
	TTP_STANDARDTITLE = 2	TTSS_NORMAL = 1 TTSS_LINK = 2
		TUS_NORMAL = 1 2 TUS_PRESSED = TUS_FOCUSED = 4 TUS_DISABLED =
TRACKBAR	TKP_THUMB = 3	
	TKP_THUMBBOTTOM = 4	TUBS_NORMAL = TUBS_HOT = 2 TUBS_PRESSED = TUBS_FOCUSED = TUBS_DISABLED =
	TKP_THUMBLEFT = 7	TUVLS_NORMAL = TUVLS_HOT = 2 TUVLS_PRESSED TUVLS_FOCUSED TUVLS_DISABLED
	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = TUVRS_HOT = 2 TUVRS_PRESSED TUVRS_FOCUSED TUVRS_DISABLED
	TKP_THUMBTOP = 5	TUTS_NORMAL = 1 TUTS_HOT = 2 TUTS_PRESSED = TUTS_FOCUSED = TUTS_DISABLED =
	TKP_THUMBVERT = 6	TUVS_NORMAL = TUVS_HOT = 2 TUVS_PRESSED = TUVS_FOCUSED = TUVS_DISABLED =

	TKP_TICS = 9	TSS_NORMAL = 1
	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	TNP_ANIMBACKGROUND = 2	
	TNP_BACKGROUND = 1	
LABELVIEW	TVP_BRANCH = 3	
		GLPS_CLOSED =
	TVP_GLYPH = 2	GLPS_OPENED =
		TREIS_NORMAL =
		TREIS_HOT = 2
		TREIS_SELECTED
	TVP_LABELITEM = 1	TREIS_DISABLED
		TREIS_SELECTED
		= 5
WINDOW	WP_CAPTION = 1	CS_ACTIVE = 1 CS
		= 2 CS_DISABLED
	WP_CAPTIONSIZINGTEMPLATE = 30	
		CBS_NORMAL = 1
	WP_CLOSEBUTTON = 18	= 2 CBS_PUSHED
		CBS_DISABLED =
	WP_DIALOG = 29	
		FS_ACTIVE = 1 FS
	WP_FRAMEBOTTOM = 9	= 2
	WP_FRAMEBOTTOMSIZINGTEMPLATE = 36	
		FS_ACTIVE = 1 FS
	WP_FRAMELEFT = 7	= 2
	WP_FRAMELEFTSIZINGTEMPLATE = 32	
		FS_ACTIVE = 1 FS
	WP_FRAMERIGHT = 8	= 2
	WP_FRAMERIGHTSIZINGTEMPLATE = 34	
		HBS_NORMAL = 1
	WP_HELPBUTTON = 23	= 2 HBS_PUSHED
		HBS_DISABLED =
		HSS_NORMAL = 1
	WP_HORZSCROLL = 25	= 2 HSS_PUSHED
		HSS_DISABLED =
		HTS_NORMAL = 1
	WP_HORZTHUMB = 26	2 HTS_PUSHED =

	HTS_DISABLED = MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_MAX_BUTTON	
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 = FS_ACTIVE = 1 FS

WP_SMALLFRAMEBOTTOM = 12		= 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37		
WP_SMALLFRAMELEFT = 10		FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMELEFTSIZINGTEMPLATE = 33		
WP_SMALLFRAMERIGHT = 11		FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35		
WP_SMALLHELPBUTTON		HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON		MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_SMALLMAXCAPTION = 6		MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4		MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON		RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON		SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13		SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_VERTSCROLL = 27		VSS_NORMAL = 1 = 2 VSS_PUSHED VSS_DISABLED =
WP_VERTTHUMB = 28		VTS_NORMAL = 1 2 VTS_PUSHED = VTS_DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

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

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

- control's **borders** using the Appearance property
- **tooltips**, Background property
- Any HTML caption that includes an <img> tag.

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

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

- control's **borders** using the Appearance property
- **tooltips**, Background property
- Any HTML caption that includes an <img> tag.

Label 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: {09ABB057-BBDE-49A6-B5CB-B05197CD337B}. The object's program identifier is: "Exontrol.Label". The /COM object module is: "ExLabel.dll"

The Exontrol's eXLabel component provides HTML labels for your forms or dialogs. A label control displays static text on a forms and allows you to manipulate it programmatically. The eXLabel control easily replaces the Standard Windows label by supporting most of the same properties, methods and events. In addition, you have complete control over how the label is to be displayed. The Label object supports the following properties and methods:

Name	Description
Alignment	Retrieves or sets the horizontal alignment of the control's caption.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoSize	Determines whether the control is automatically resized to fit its entire content.
BackColor	Specifies the control's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BackgroundExt	Indicates additional colors, text, images that can be displayed on the object's background using the EBN string format.
BackgroundExtValue	Specifies at runtime, the value of the giving property for specified part of the background extension.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BorderHeight	Sets or retrieves a value that indicates the border height of the control.
BorderWidth	Sets or retrieves a value that indicates the border width of the control.
Caption	Specifies the control's caption.
Enabled	Enables or disables the control.
	Resumes painting the control after painting is suspended

[EndUpdate](#)	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.
[Font](#)	Retrieves or sets the control's font.
[ForeColor](#)	Specifies the control's foreground color.
[FormatAnchor](#)	Specifies the visual effect for anchor elements in HTML captions.
[HTMLPicture](#)	Adds or replaces a picture in HTML captions.
[hWnd](#)	Retrieves the control's window handle.
[Images](#)	Sets at runtime the control's image list. The Handle should be a handle to an Images List Control.
[ImageSize](#)	Retrieves or sets the size of icons the control displays..
[Picture](#)	Retrieves or sets a graphic to be displayed in the control.
[PictureDisplay](#)	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
[Refresh](#)	Refreshes the control.
[ReplaceIcon](#)	Adds a new icon, replaces an icon or clears the control's image list.
[RequiredHeight](#)	Returns the height of the label required to display its entire contents.
[RequiredWidth](#)	Returns the width of the label required to display its entire contents.
[Rotate](#)	Rotates the HTML caption.
[ShowImageList](#)	Specifies whether the control's image list window is visible or hidden.
[ShowToolTip](#)	Shows the specified tooltip at given position.
[Template](#)	Specifies the control's template.
[TemplateDef](#)	Defines inside variables for the next Template/ExecuteTemplate call.
[TemplatePut](#)	Defines inside variables for the next Template/ExecuteTemplate call.
[ToolTipDelay](#)	Specifies the time in ms that passes before the ToolTip appears.

[ToolTipFont](#)	Retrieves or sets the tooltip's font.
[ToolTipPopDelay](#)	Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.
[ToolTipText](#)	Specifies the control's tooltip text.
[ToolTipTitle](#)	Specifies the title of the control's tooltip.
[ToolTipWidth](#)	Specifies a value that indicates the width of the tooltip window, in pixels.
[Transparent](#)	Indicates whether the control's background is transparent.
[VAlignment](#)	Retrieves or sets the vertical alignment of the control's caption.
[Version](#)	Retrieves the control's version.
[VisualAppearance](#)	Retrieves the control's appearance.
[WordWrap](#)	Indicates whether a multiline label control automatically wraps words to the beginning of the next line when necessary.

property Label.Alignment as AlignmentEnum

Retrieves or sets the alignment of the control's caption.

Type	Description
[AlignmentEnum](#)	An AlignmentEnum expression that indicates the caption's alignment in the label.

By default, the caption is left aligned. Use the Alignment property to align the caption to the right, or center. Use the [VAlignment](#) property to specify the vertical alignment of the caption inside the label. Use the [Caption](#) property to assign a caption to the label. Use the <r> HTML element to align a portion of the label to the right.

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

Retrieves the identifier of the anchor from point.

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

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

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

```
Private Sub Label1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   With Label1
      .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 AxLabel1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXLABELLib._ILabelEvents_MouseMoveEvent) Handles AxLabel1.MouseMoveEvent
   With AxLabel1
      .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:

```csharp
private void axLabel1_MouseMoveEvent(object sender,
AxEXLABELLib._ILabelEvents_MouseMoveEvent e)
{
    axLabel1.ShowToolTip(axLabel1.get_AnchorFromPoint(-1, -1));
}
```

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

```cpp
void OnMouseMoveLabel1(short Button, short Shift, long X, long Y)
{
    COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
    m_label.ShowToolTip( m_label.GetAnchorFromPoint( -1, -1 ), vtEmpty, vtEmpty, vtEmpty );
}
```

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

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

with thisform
    With .Label1
        .ShowToolTip(.AnchorFromPoint(-1, -1))
    EndWith
endwith
```

property Label.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the control's appearance, or a color expression whose last 7 bits in the high significant byte of the value indicates the index of the skin in the Appearance collection, being displayed as control's borders. For instance, if the Appearance = 0x1000000, indicates that the first skin object in the Appearance collection defines the control's border. *The Client object in the skin, defines the client area of the control. The caption is always shown in the label's client area. The skin may contain transparent objects, and so you can define round corners. The frame.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file*

By default, the control shows no borders. Use the Appearance property to specify the control's border. Use the Add method to add new skins to the control. Use the BackColor property to specify the control's background color. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the BorderWidth, BorderHeight property to indent the control's label based on the control's client area. The RequiredHeight property returns the height of the label required to display its entire contents. The RequiredWidth property returns the width of the label required to display its entire contents.
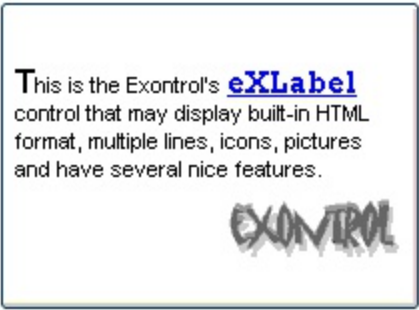
The following VB sample changes the visual aspect of the borders of the control (please check the above picture for round corners):

```
With Label1
    .BeginUpdate
        .VisualAppearance.Add &H16, "c:\temp\frame.ebn"
```

```
        .Appearance = &H16000000
        .BackColor = RGB(250, 250, 250)
    .EndUpdate
End With
```

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

```
With AxLabel1
    .BeginUpdate()
    .VisualAppearance.Add(&H16, "c:\temp\frame.ebn")
    .Appearance = &H16000000
    .BackColor = Color.FromArgb(250, 250, 250)
    .EndUpdate()
End With
```

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

```
axLabel1.BeginUpdate();
axLabel1.VisualAppearance.Add(0x16, "c:\\temp\\frame.ebn");
axLabel1.Appearance = (EXLABELLib.AppearanceEnum)0x16000000;
axLabel1.BackColor = Color.FromArgb(250, 250, 250);
axLabel1.EndUpdate();
```

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

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

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

```
with thisform.Label1
    .BeginUpdate
        .VisualAppearance.Add(0x16, "c:\temp\frame.ebn")
        .Appearance = 0x16000000
        .BackColor = RGB(250, 250, 250)
    .EndUpdate
```

endwith

method Label.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 Label1_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 Label.AutoSize as Boolean

Determines whether the control is automatically resized to fit its entire content.

Type	Description
Boolean	A Boolean expression that specifies whether the control is automatically resized to fit its entire content.

By default, the AutoSize property is False. The AutoSize property resizes the control to fit its entire content. Use the Caption property to specify the control's label. The AutoSize property has effect only when calling, so if you change the control's label at runtime, the control will not be resized, until you call again the AutoSize property. So, call the AutoSize property only when required. Use the WordWrap property to determine whether the control expands to fit the caption in the client's area. If the WordWrap property is True, and AutoSize property is called on True, the control's uses the original width to compute only the height of the control. The AutoSize property may be useful for single-line labels. The RequiredHeight property returns the height of the label required to display its entire contents. The RequiredWidth property returns the width of the label required to display its entire contents.

property Label.BackColor as Color

Specifies the control's background color.

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

The BackColor property assigns a different background color for the entire label. Use the Picture property to assign a picture on the control's background. Use the Appearance property to change the control's border. Use the ForeColor property to assign a different foreground color for label's caption. Use the <bgcolor> HTML element to specify a different background color for portions of text in the control's Caption. Use the HTMLPicture property to add custom size pictures, and display them in the control's caption using the <img> HTML element. Use the Transparent property on True, to display the label using a transparent background color, so the parent window is responsible to provide the label's background.

property Label.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](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](Add) method to add new skins to the control. Use the Remove method to remove a specific skin from the control. Use the Clear method to remove all skins in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while init the control. Use the Refresh method to refresh the control.

The following VB sample changes the visual appearance of the control's tooltip. The sample applies the frame.ebn skin to the borders of the control's tooltip:

```
With Label1
   With .VisualAppearance
      .Add &H1, App.Path + "\frame.ebn"
   End With
   .Background(exToolTipAppearance) = &H1000000
End With
```

The following C++ sample changes the visual appearance for the "drop down" filter button:

```
#include "Appearance.h"
m_label.GetVisualAppearance().Add( 0x01,
COleVariant(_T("D:\\Temp\\ExLabel.Help\\frame.ebn")) );
m_label.SetBackground( 0 /*exToolTipAppearance*/, 0x1000000 );
```

The following VB.NET sample changes the visual appearance for the "drop down" filter button:

```
With AxLabel1
    With .VisualAppearance
        .Add(&H1, "D:\Temp\ExLabel.Help\frame.ebn")
    End With
    .set_Background(EXLABELLib.BackgroundPartEnum.exToolTipAppearance, &H1000000)
End With
```

The following C# sample changes the visual appearance for the "drop down" filter button:

```
axLabel1.VisualAppearance.Add(0x1, "D:\\Temp\\ExLabel.Help\\frame.ebn");
axLabel1.set_Background(EXLABELLib.BackgroundPartEnum.exToolTipAppearance,
0x1000000);
```

The following VFP sample changes the visual appearance for the "drop down" filter button:

```
With thisform.Label1
    With .VisualAppearance
        .Add(1, "D:\Temp\ExLabel.Help\frame.ebn")
    EndWith
    .Object.Background(64) = 16777216
EndWith
```

The 16777216 value is the 0x1000000 value in hexadecimal.

property Label.BackgroundExt as String

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

Type	Description
String	A String expression (**"EBN String Format"**) that defines the layout of the UI to be applied on the object's background. The syntax of EBN String Format in BNF notation is shown bellow. *You can use the EBN's Builder of eXLabel/COM control to define visually the EBN String Format.*

By default, the BackgroundExt property is empty. Using the BackgroundExt property you have unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the object's background. *For instance, let's say you need to display more colors on the object's background, or just want to display an additional caption or image to a specified location on the object's background.* The EBN String Format defines the parts of the EBN to be applied on the object's background. The EBN is a set of UI elements that are built as a tree where each element is anchored to its parent element. Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExt property is applied right after setting the object's backcolor, and before drawing the default object's captions, icons or pictures.
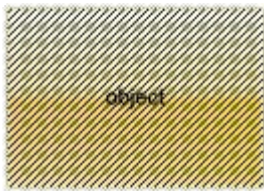
Complex samples:
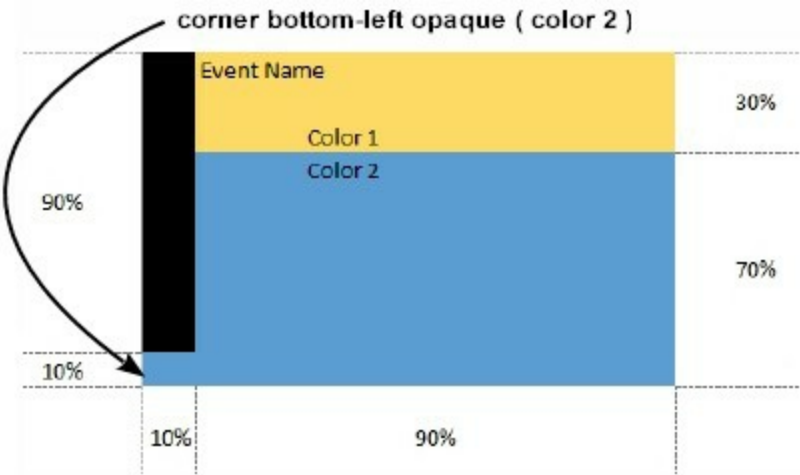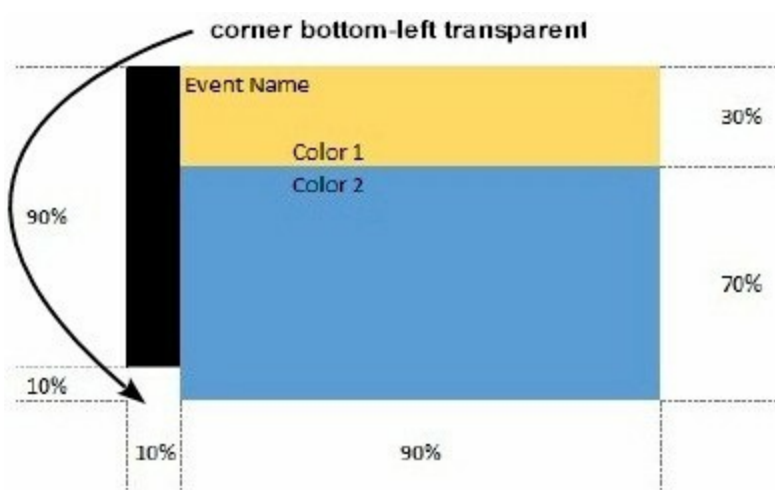
Easy samples:

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

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


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


- "[[patterncolor=RGB(255,0,0)]
  (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25
  draws a red thick-border around the object, with a patter inside, with a 4-pixels wide
  padding:
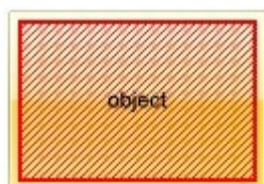
- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.
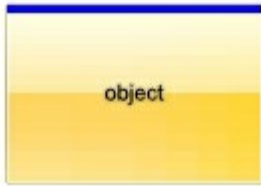
- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.
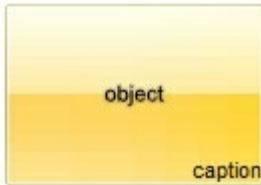
- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.
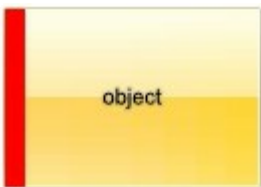
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.

- "root[text=`caption <b>2`,align=0x22](client[text=`caption <b>1`,align=0x20])", shows the caption **1** aligned to the bottom-left side, and the caption **2** to the bottom-right side
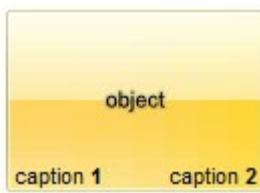
*The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field as shown in the following screen shot:*

The **To String** field of the EBN Builder defines the **EBN String Format** that can be used on BackgroundExt property.
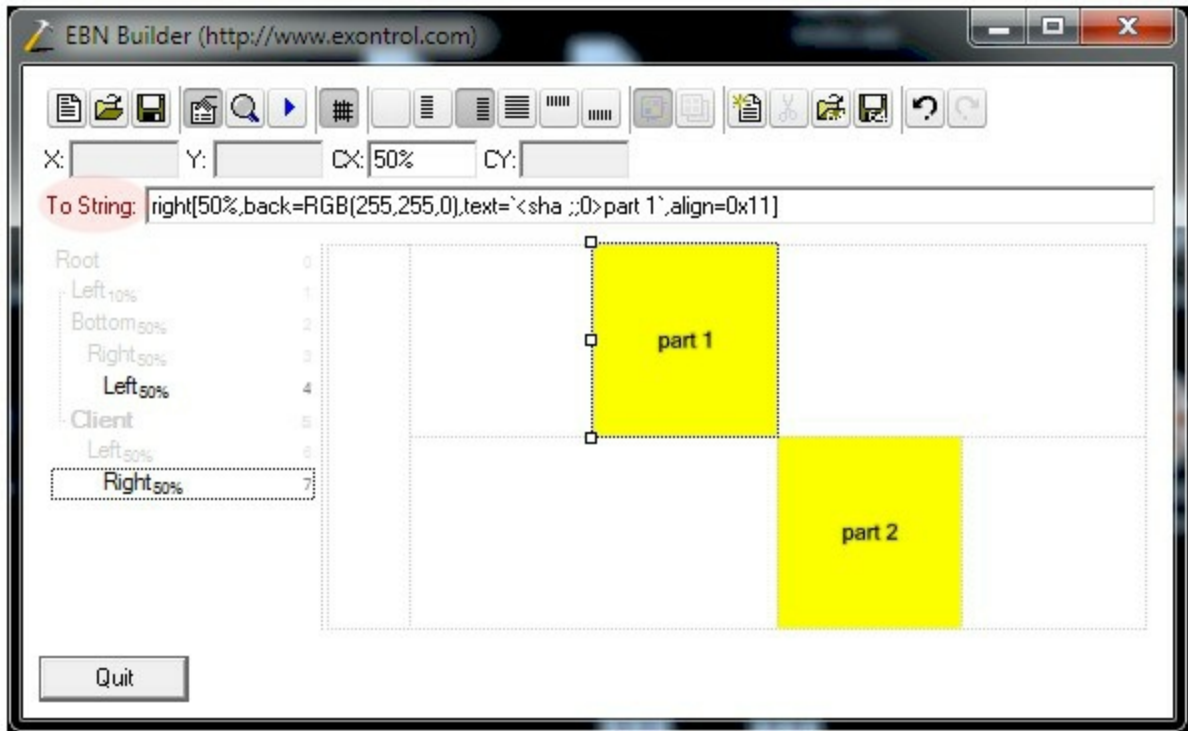
The **EBN String Format** syntax in BNF notation is defined like follows:

```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<client> ","] <attribute> [ "," <attributes> ] "]"
<client> ::= <expression> | <expression> "," <expression> "," <expression> ","
<expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> | <wordwrap> | <align> | <pattern> |
<patterncolor> | <frame> | <framethick> | <data> | <others>
<equal> ::= "="
```

```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B" "C" | "D" | "E" "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number> ")"
<string> ::= "`" <characters> "`" | "'" <characters> "'" | " <characters> "
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
<patterncolor> ::= "patterncolor" <equal> <color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

Now, lets say we have the following request to layout the colors on the objects:
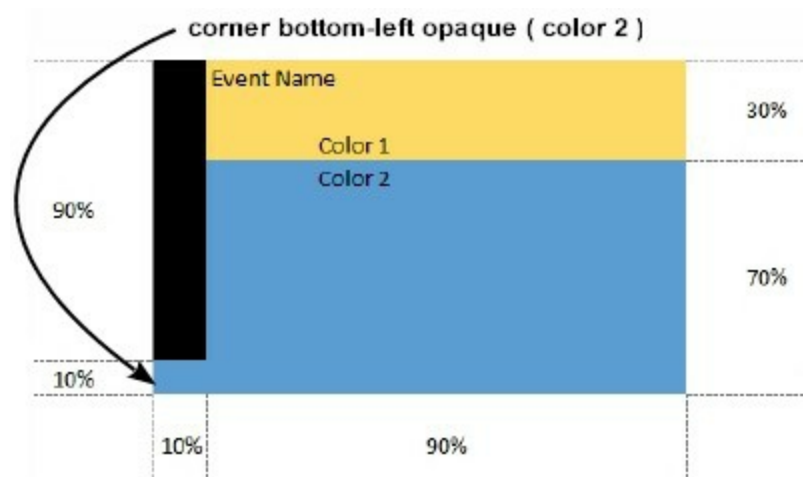
We define the BackgroundExt property such as
"top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100%
(top[90%,back=RGB(0,0,0)])]", and it looks as:

To String: 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)]]

Root
  Top 30%   1
  Client   2
  None 0%,0% 10%,100%   3
      Top 90%   4

so, if we apply to our object we got:

object

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

corner bottom-left transparent

Event Name
                                    30%
Color 1
Color 2
90%
                                    70%

10%

10%   90%
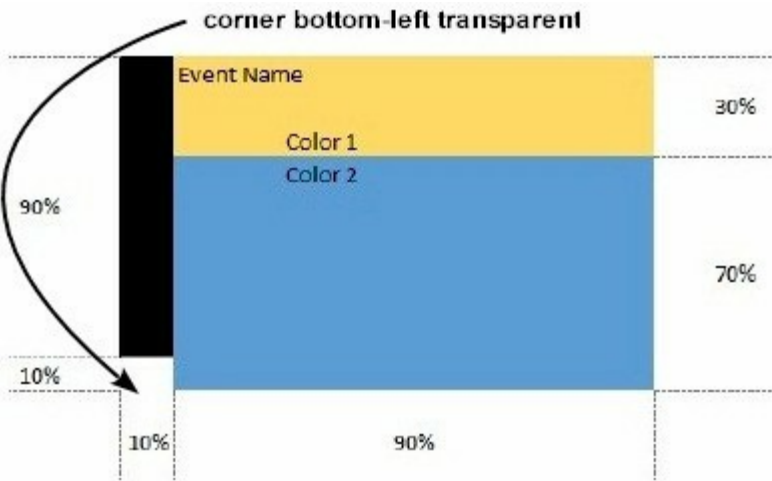
We define BackgroundExt property such as "left[10%]
(top[90%,back=RGB(0,0,0)]]),top[30%,back=RGB(254,217,102)],client[back=RGB(91,156,2
and it looks as:

To String: left[10%][top[90%,back=RGB(0,0,0)]],top[30%,back=RGB(254,217,102)],client[back=RGB(91,156,212)]

Root
  Left 10%   1
      Top 90%   2
  Top 30%   3
  Client   4
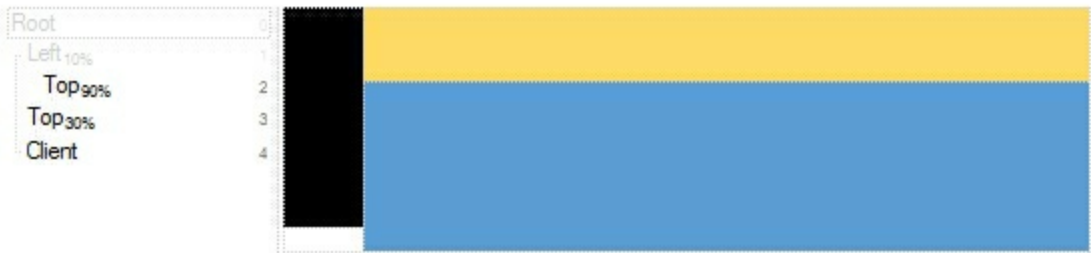
so, if we apply to our object we got:

object
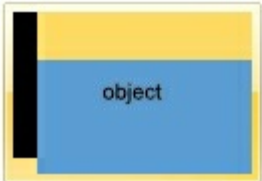
property Label.BackgroundExtValue(Index as IndexExtEnum, Property as BackgroundExtPropertyEnum) as Variant

Specifies at runtime, the value of the giving property for specified part of the background extension.

Type	Description
Index as IndexExtEnum	A Long expression that defines the index of the part that composes the EBN to be accessed / changed. The following screen shot shows where you can find Index of the parts: The screen shot shows that the EBN contains 11 elements, so in this case the Index starts at 0 (root element) and ends on 10.
Property as BackgroundExtPropertyEnum	A BackgroundExtPropertyEnum expression that specifies the property to be changed as explained bellow.
Variant	A Variant expression that defines the part's value. The Type of the expression depending on the Property parameter as explained bellow.
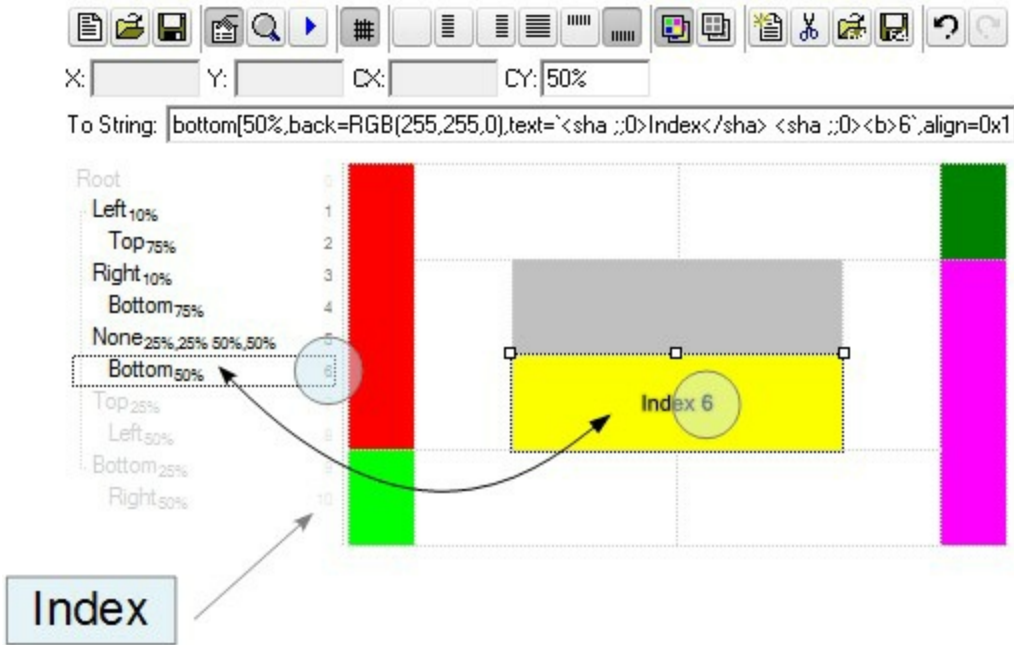
Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExtValue property has no effect if the BackgroundExt property is empty (by default). *The idea is as follows: first you need to decide the layout of the UI to put on the object's background, using the BackgroundExt*

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 BackgroundExt 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 BackgroundExt on "bottom[50%,pattern=6,frame]"

we got:
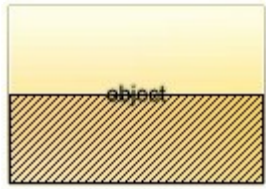
so let's change the percent of 50% to 25% like BackgroundExtValue(1,2) on "25%", where 1 indicates the first element after root, and 2 indicates the **exClientExt** property, we get:
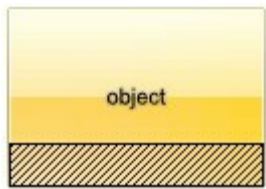
In VB you should have the following syntax:

```
.BackgroundExt = "bottom[50%,pattern=6,frame]"
.BackgroundExtValue(exIndexExt1, exClientExt) = "25%"
```

method Label.BeginUpdate ()

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

Type	Description

The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Use the Refresh method to force refreshing the control.

property Label.BorderHeight as Long

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

Type	Description
Long	A long expression that specifies the width in pixels, to indent the control's label from the top and bottom borders.

By default, the BorderHeight property is 0. Use the [BorderWidth](#) and BorderHeight property to indent the control's label from the borders of the control. Use the [Appearance](#) property to specify the control's borders. Use the [AutoSize](#) property to resize the control to fit its entire content.

property Label.BorderWidth as Long

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

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

By default, the BorderWidth property is 0. Use the BorderWidth and [BorderHeight](#) property to indent the control's label from the borders of the control. Use the [Appearance](#) property to specify the control's borders. Use the [AutoSize](#) property to resize the control to fit its entire content. The [RequiredHeight](#) property returns the height of the label required to display its entire contents. The [RequiredWidth](#) property returns the width of the label required to display its entire contents.

property Label.Caption as String

Specifies the control's caption.

Type	Description
String	A String expression that indicates the caption being displayed in the label. The string may contains built-in HTML element like explained.

The Caption property specifies the label's caption. Use the Images method to assign a list of icons being displayed in the caption using the <img> HTML element. Use the HTMLPicture property to add new custom size pictures to be displayed in the label, using the <img> HTML element. Use the ToolTipText property to assign a tooltip to the label. The AnchorClick event notifies your application when the user clicks an anchor/link element. Use the VAlignment property to specify the vertical alignment of the caption inside the label. Use the Alignment property to specify the horizontal alignment of the caption inside the label. If the control's AutoSize property is True, the control is automatically resized to fit its entire content.

The Caption property supports the following built-in HTML elements:

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

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

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY

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

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

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=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:

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

Use the Appearance property to specify the control's border. Use the BorderWidth, BorderHeight property to indent the control's label based on the control's client area. Use the BackColor property to specify the label's background color. Use the ForeColor property to specify the control's foreground color. The RequiredHeight property returns the height of the label required to display its entire contents. The RequiredWidth property returns the width of the label required to display its entire contents.

property Label.Enabled as Boolean

Enables or disables the control.

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

By default, the Enabled property is False. Use the Enabled property to disable the label. A disabled label displays its content in gray. While the label is disabled, no mouse or keys events are fired. Use the hWnd property to retrieve the handle of the control's window.

method Label.EndUpdate ()

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

Type	Description

The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Use the Refresh method to force refreshing the control.

property Label.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 Label.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 label's background color:

```
Debug.Print Label1.ExecuteTemplate("BackColor")
```

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

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

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

The Template script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters.

An instruction can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property(list of arguments) = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method(list of arguments) *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)**....** *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The Template supports the following general functions:

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier*

property Label.Font as IFontDisp

Retrieves or sets the control's font.

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

Use the Font property to change the control's font. Use the <b>,<i>, <s> or <u> HTML attributes in the [Caption](#) property to show portions of text as bold, italic, strikeout or underlined. Use the <font> HTML element to specify a different font for portions on text in the control's caption.

property Label.ForeColor as Color

Specifies the control's foreground color.

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

The ForeColor property assigns a different foreground color for the entire label. Use the Picture property to assign a picture on the control's background. Use the Appearance property to change the control's border. Use the BackColor property to assign a different background color for label's caption. Use the <fgcolor> HTML element to specify a different foreground color for portions of text in the control's Caption.

property Label.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 Label.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 Images method to assign a list of icons being displayed in the Caption property.

The following sample shows how to put a custom size picture in the label:

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

<CONTROL>.Caption = "A <img>pic1</img>"
<COLUMN2>.Caption = "B <img>pic2</img>"
<COLUMN3>.Caption = "A <img>pic1</img> + B <img>pic2</img>"
```

property Label.hWnd as Long

Retrieves the control's window handle.

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

Use the hWnd property to get the handle of the control's main 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 Label.Images (Handle as Variant)

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

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

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

The user can add images at design time, by drag and drop files to combo's image holder. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ShowImageList property to show or hide the control's images panel at design mode. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. Use the <img> HTML element in the Caption property to display icons in your label. Use the HTMLPicture property to add custom size pictures to be displayed in your label, using the <img> HTML element.

property Label.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 Label.Picture as IPictureDisp

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

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

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the BackColor property to specify the control's background color. Use the Appearance property to change the control's border. Use the BorderWidth, BorderHeight property to indent the control's label based on the control's client area. Use the HTMLPicture property to add custom size pictures, and display them in the control's Caption using the <img> HTML element.

property Label.PictureDisplay as PictureDisplayEnum

Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background

Type	Description
[PictureDisplayEnum](#)	A PictureDisplayEnum expression that indicates the way how the picture is displayed.

By default, the PictureDisplay property is exTile. Use the PictureDisplay property specifies how the [Picture](#) is displayed on the control's background. If the control has no picture associated the PictureDisplay property has no effect.

method Label.Refresh ()

Refreshes the control.

Type	Description

The Refresh method forces repainting the control. Use the BeginUpdate and EndUpdate methods to maintain the control's performance while adding multiple items or columns. Use the hWnd property to get the handle of the control's window.

The following VB sample calls the Refresh method:

```
Label1.Refresh
```

The following C++ sample calls the Refresh method:

```
m_Label.Refresh();
```

The following VB.NET sample calls the Refresh method:

```
AxLabel1.CtlRefresh()
```

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

The following C# sample calls the Refresh method:

```
axLabel1.CtlRefresh();
```

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

The following VFP sample calls the Refresh method:

```
thisform.Label1.Object.Refresh()
```

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

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

Type	Description
Icon as Variant	A long expression that indicates the icon's handle.
Index as Variant	A long expression that indicates the index where icon is inserted.

Return	Description
Long	A long expression that indicates the index of the icon in the images collection.

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

The following VB sample adds a new icon to control's images list:

 i = ExLabel1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added

The following VB sample replaces an icon into control's images list::

 i = ExLabel1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.

The following VB sample removes an icon from control's images list:

 ExLabel1.ReplaceIcon 0, i, i specifies the index of icon removed.

The following VB clears the control's icons collection:

 ExLabel1.ReplaceIcon 0, -1

property Label.RequiredHeight as Long

Returns the height of the label required to display its entire contents.

Type	Description
Long	A long expression that specifies the height of the label required to display its entire contents.

The RequiredHeight property is read only. The RequiredHeight property gets the height required to resize the control's content so its caption fits entirely. The RequiredWidth property returns the width of the label required to display its entire contents. If the control's AutoSize property is True, the control is automatically resized to fit its entire content. The RequiredHeight property includes the control's borders.

property Label.RequiredWidth as Long

Returns the width of the label required to display its entire contents.

Type	Description
Long	A long expression that specifies the width of the label required to display its entire contents.

The RequiredWidth property is read only. The RequiredWidth property gets the width required to resize the control's content so its caption fits entirely. The RequiredHeight property returns the height of the label required to display its entire contents. If the control's AutoSize property is True, the control is automatically resized to fit its entire content. The RequiredWidth property includes the control's borders.

property Label.Rotate as HTMLRotateEnum

Rotates the HTML caption.

Type	Description
HTMLRotateEnum	A HTMLRotateEnum expression that specifies how the control displays the HTML caption.

By default, the Rotate property is exHTMLHorizontal, which indicates that the text is displayed horizontally. The Rotate property rotates or displays in mirror the HTML caption. The Rotate property can be one of the following:

- **exHTMLHorizontal**, displays horizontally the caption
- **exHTMLHorizontal + exHTMLMirror**, displays horizontally the caption, in the mirror
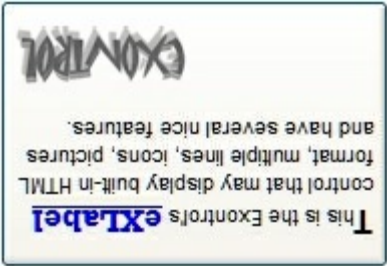
- **exHTMLVertical**, displays vertically the caption
- **exHTMLVertical + exHTMLMirror**, displays vertically the caption, in the mirror

property Label.ShowImageList as Boolean

Specifies whether the control's image list window is visible or hidden.

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

By default, the ShowImageList property is True. Use the ShowImageList property to hide the control's images list window. The control's images list window is visible only at design time. Use the Images method to associate an images list control to the Label control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection.

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

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

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

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

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified **background** color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the [Add](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:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Label.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 gets the result after executing a template script.

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

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

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

The Template script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters.

An instruction can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values*

separated by commas. (Sample: h = InsertItem(0,"New Child"))
- property(list of arguments) = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method(list of arguments) *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)**.**... *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The Template supports the following general functions:

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property Label.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 Label.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 Label.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 ToolTipText property to specify the label's tooltip. Use the ShowToolTip method to display a custom tooltip.

property Label.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

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

property Label.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 ToolTipText property to specify the label's tooltip. Use the ShowToolTip method to display a custom tooltip.

property Label.ToolTipText as String

Specifies the control's tooltip text.

Type	Description
String	A string expression that defines the label's tooltip

Use the ToolTipText and ToolTipTitle properties to define the label's tooltip. Use the ToolTipDelay and ToolTipPopDelay properties to specify the time in ms that passes before the ToolTip appears. Use the **<img>** HTML tag to insert icons or custom size pictures inside the label's tooltip. Use the ShowToolTip method to display programmatically a custom tooltip.

The ToolTipText supports the following HTML tags:

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

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

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

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

**
Line1<r><a ;exp=show lines>+
Line2
Line3"** shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified <mark style="background-color:red;color:white">background</mark> 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 **&amp;** (&), **&lt;** (<), **&gt;** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the &#8364; 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:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

In the VB environment, the extended wrapper control that implements properties like Visible, Top, Left, Width, Height ... and so on includes also a property named TooltipText that can provide confusions. In order to avoid confusions on this name, the VB users must call at runtime a code like:

```
With Label1
    .Object.ToolTipText = "In VB, this is the right way to call the ToolTipText property at runtime."
End With
```

If the Object property is missing, the code changes the TooltipText property of the VB extended class, instead calling object's property.

property Label.ToolTipTitle as String

Specifies the title of the control's tooltip.

Type	Description
String	A String expression that defines the control's tooltip title.

Use the ToolTipText and ToolTipTitle properties to define the Label's tooltip. Use the ToolTipDelay and ToolTipPopDelay properties to specify the time in ms that passes before the ToolTip appears. Use the ShowToolTip method to display programmatically a custom tooltip.

property Label.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 ToolTipText property to specify the label's tooltip. Use the ShowToolTip method to display a custom tooltip.

property Label.Transparent as Boolean

Indicates whether the control's background is transparent.

Type	Description
Boolean	A Boolean expression that specifies whether the control's background is opaque or transparent

By default, the Transparent property is False. Use the Transparent property on True, to display the label using a transparent background color, so the parent window is responsible to provide the label's background. The BackColor property of the control has no effect if the Transparent property is True. Use the HTMLPicture property to add custom size pictures, and display them in the control's caption using the <img> HTML element. The Caption property specifies the label's caption. For instance, in VB6 you can use the form's Picture property to put a picture on the form's background, and so it will be displayed on the Label's background too, if the Transparent property is True. In /NET framework, the BackgroundImage property can be used to specify the form's background image.

The following screen show shows the control with a Transparent background (Transparent property is True):

The following screen show shows the control with an opaque background (Transparent property is False, by default):

property Label.VAlignment as VAlignmentEnum

Retrieves or sets the vertical alignment of the control's caption.

Type	Description
VAlignmentEnum	Specifies the caption's vertical alignment.

By default, the VAlignment property is MiddleAlignment. Use the VAlignment property to vertically align the caption inside the label. Use the Alignment property to specify the horizontal alignment of the caption inside the label. Use the Caption property to specify the label's caption. Use the WordWrap property to specify whether the label wraps the caption.

property Label.Version as String

Retrieves the control's version.

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

The version property specifies the control's version.

property Label.VisualAppearance as Appearance

Retrieves the control's appearance.

Type	Description
Appearance	An Appearance object that holds a collection of skins.

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part.
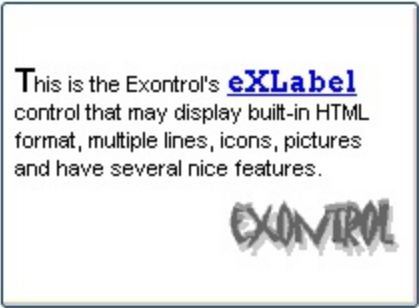
The skin method may change the visual appearance for the following parts in the control:

- control's **borders** using the Appearance property
- **tooltips**, Background property
- Any HTML caption that includes an <img> tag.

property Label.WordWrap as Boolean

Indicates whether a multiline label control automatically wraps words to the beginning of the next line when necessary.

Type	Description
Boolean	A Boolean expression that specifies whether the control automatically wraps words to the beginning of the next line when necessary.

By default, the WordWrap property is False. If the WordWrap property is True, the control wraps words. Use the <br> to break programmatically a line. Use the Caption property to assign a HTML caption to your label.

ExLabel events

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {09ABB057-BBDE-49A6-B5CB-B05197CD337B}. The object's program identifier is: "Exontrol.Label". The /COM object module is: "ExLabel.dll"

The ExLabel component supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user presses and then releases the left mouse button over the control.
DblClick	Occurs when the user dblclk the left mouse button over an object.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.

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,
AxEXLABELLib._ILabelEvents_AnchorClickEvent e)
{
}
```

```cpp
void OnAnchorClick(LPCTSTR AnchorID,LPCTSTR Options)
{
}
```

```cpp
void __fastcall AnchorClick(TObject *Sender,BSTR AnchorID,BSTR Options)
{
}
```

```delphi
procedure AnchorClick(ASender: TObject; AnchorID : WideString;Options :
WideString);
begin
end;
```

```delphi
procedure AnchorClick(sender: System.Object; e:
AxEXLABELLib._ILabelEvents_AnchorClickEvent);
begin
end;
```

```
begin event AnchorClick(string AnchorID,string Options)
end event AnchorClick
```

```vbnet
Private Sub AnchorClick(ByVal sender As System.Object, ByVal e As
AxEXLABELLib._ILabelEvents_AnchorClickEvent) Handles AnchorClick
End Sub
```

```vb
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
```

```
LPARAMETERS AnchorID,Options
```

```
PROCEDURE OnAnchorClick(oLabel,AnchorID,Options)
RETURN
```

Syntax for AnchorClick event, **/COM** version ^(others), on:

Java…	`<SCRIPT EVENT="AnchorClick(AnchorID,Options)" LANGUAGE="JScript">` `</SCRIPT>`

VBSc…	`<SCRIPT LANGUAGE="VBScript">`
	`Function AnchorClick(AnchorID,Options)`
	`End Function`
	`</SCRIPT>`

Visual Data…	`Procedure OnComAnchorClick String llAnchorID String llOptions`
	`    Forward Send OnComAnchorClick llAnchorID llOptions`
	`End_Procedure`

Visual Objects	`METHOD OCX_AnchorClick(AnchorID,Options) CLASS MainDialog` `RETURN NIL`

X++	`void onEvent_AnchorClick(str _AnchorID,str _Options)`
	`{`
	`}`

XBasic	`function AnchorClick as v (AnchorID as C,Options as C)` `end function`

dBASE	`function nativeObject_AnchorClick(AnchorID,Options)` `return`

event Click ()

Occurs when the user presses and then releases the left mouse button over the control.

Type	Description

The Click event is fired when the user releases the left mouse button over the control. Use a [MouseDown](#) or [MouseUp](#) event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and [DblClick](#) events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

Syntax for Click event, **/NET** version, on:

```csharp
C#   private void Click(object sender)
     {
     }
```

```vb
VB   Private Sub Click(ByVal sender As System.Object) Handles Click
     End Sub
```

Syntax for Click event, **/COM** version, on:

```csharp
C#   private void ClickEvent(object sender, EventArgs e)
     {
     }
```

```cpp
C++   void OnClick()
      {
      }
```

```cpp
C++
Builder   void __fastcall Click(TObject *Sender)
          {
          }
```

```delphi
Delphi   procedure Click(ASender: TObject; );
         begin
         end;
```

```
procedure ClickEvent(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe...
```
begin event Click()
end event Click
```

VB.NET
```
Private Sub ClickEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ClickEvent
End Sub
```

VB6
```
Private Sub Click()
End Sub
```

VBA
```
Private Sub Click()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas...
```
PROCEDURE OnClick(oLabel)
RETURN
```

Syntax for Click event, **/COM** version (others), on:

Java...
```
<SCRIPT EVENT="Click()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc...
```
<SCRIPT LANGUAGE="VBScript">
Function Click()
End Function
</SCRIPT>
```

Visual Data...
```
Procedure OnComClick
    Forward Send OnComClick
```

End_Procedure

METHOD OCX_Click() CLASS MainDialog
RETURN NIL

```
void onEvent_Click()
{
}
```

```
function Click as v ()
end function
```

```
function nativeObject_Click()
return
```

event DblClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user dblclk the left mouse button over an object.

Type	Description
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The DblClick event is fired when user double clicks the control. Use the AnchorFromPoint property to get the anchor element from the point.

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, AxEXLABELLib._ILabelEvents_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:`
`AxEXLABELLib._ILabelEvents_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`
`AxEXLABELLib._ILabelEvents_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(oLabel,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

```
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.Label.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Label.1::OLE_YPOS_PIXELS)
end function
```

dBASE	

```
function nativeObject_DblClick(Shift,X,Y)
return
```

event KeyDown (KeyCode as Integer, Shift as Integer)

Occurs when the user presses a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and [KeyUp](#) event procedures if you need to respond to both the pressing and releasing of a key. You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the And operator with the shift argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

Syntax for KeyDown event, **/NET** version, on:

```csharp
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:

```csharp
private void KeyDownEvent(object sender,
AxEXLABELLib._ILabelEvents_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:
AxEXLABELLib._ILabelEvents_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
AxEXLABELLib._ILabelEvents_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(oLabel,KeyCode,Shift)
RETURN
```

Syntax for KeyDown event, **/COM** version ^(others), on:

| Java… | ```<SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">```
 ```</SCRIPT>``` |

```
<SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

```
<SCRIPT LANGUAGE="VBScript">
Function KeyDown(KeyCode,Shift)
End Function
</SCRIPT>
```

```
Procedure OnComKeyDown Short llKeyCode Short llShift
    Forward Send OnComKeyDown llKeyCode llShift
End_Procedure
```

```
METHOD OCX_KeyDown(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

```
void onEvent_KeyDown(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

```
function KeyDown as v (KeyCode as N,Shift as N)
end function
```

```
function nativeObject_KeyDown(KeyCode,Shift)
return
```

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	An integer that returns a standard numeric ANSI keycode.

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress does not indicate the physical state of the keyboard; instead, it passes a character. KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters.

Syntax for KeyPress event, **/NET** version, on:

```C#
private void KeyPress(object sender,ref short KeyAscii)
{
}
```

```VB
Private Sub KeyPress(ByVal sender As System.Object,ByRef KeyAscii As Short)
Handles KeyPress
End Sub
```

Syntax for KeyPress event, **/COM** version, on:

```C#
private void KeyPressEvent(object sender,
AxEXLABELLib._ILabelEvents_KeyPressEvent e)
{
}
```

```C++
void OnKeyPress(short FAR* KeyAscii)
{
}
```

```C++ Builder
void __fastcall KeyPress(TObject *Sender,short * KeyAscii)
{
}
```

Delphi	

	`procedure KeyPress(ASender: TObject; var KeyAscii : Smallint);`
	`begin`
	`end;`

Delphi 8 (.NET only)	

	`procedure KeyPressEvent(sender: System.Object; e:`
	`AxEXLABELLib._ILabelEvents_KeyPressEvent);`
	`begin`
	`end;`

Powe…	
	`begin event KeyPress(integer KeyAscii)` `end event KeyPress`

VB.NET	

	`Private Sub KeyPressEvent(ByVal sender As System.Object, ByVal e As`
	`AxEXLABELLib._ILabelEvents_KeyPressEvent) Handles KeyPressEvent`
	`End Sub`

VB6	
	`Private Sub KeyPress(KeyAscii As Integer)` `End Sub`

VBA	
	`Private Sub KeyPress(KeyAscii As Integer)` `End Sub`

VFP	
	`LPARAMETERS KeyAscii`

Xbas…	
	`PROCEDURE OnKeyPress(oLabel,KeyAscii)` `RETURN`

Syntax for KeyPress event, **/COM** version (others), on:

Java…	
	`<SCRIPT EVENT="KeyPress(KeyAscii)" LANGUAGE="JScript">` `</SCRIPT>`

VBSc…	

	`<SCRIPT LANGUAGE="VBScript">`
	`Function KeyPress(KeyAscii)`
	`End Function`
	`</SCRIPT>`

Visual Data...	Procedure OnComKeyPress Short llKeyAscii Forward Send OnComKeyPress llKeyAscii End_Procedure

Visual Objects	METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog RETURN NIL

X++	void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii)
	{
	}

XBasic	function KeyPress as v (KeyAscii as N) end function

dBASE	function nativeObject_KeyPress(KeyAscii) return

event KeyUp (KeyCode as Integer, Shift as Integer)

Occurs when the user releases a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the KeyUp event procedure to respond to the releasing of a key.

Syntax for KeyUp event, **/NET** version, on:

```C#
private void KeyUp(object sender,ref short KeyCode,short Shift)
{
}
```

```VB
Private Sub KeyUp(ByVal sender As System.Object,ByRef KeyCode As Short,ByVal Shift As Short) Handles KeyUp
End Sub
```

Syntax for KeyUp event, **/COM** version, on:

```C#
private void KeyUpEvent(object sender, AxEXLABELLib._ILabelEvents_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:
AxEXLABELLib._ILabelEvents_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
AxEXLABELLib._ILabelEvents_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(oLabel,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>

```
Procedure OnComKeyUp Short llKeyCode Short llShift
    Forward Send OnComKeyUp llKeyCode llShift
End_Procedure
```

```
METHOD OCX_KeyUp(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

```
void onEvent_KeyUp(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

```
function KeyUp as v (KeyCode as N,Shift as N)
end function
```

```
function nativeObject_KeyUp(KeyCode,Shift)
return
```

event MouseDown (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user presses a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

Use a MouseDown or [MouseUp](#) event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the [Click](#) and [DblClick](#) events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the [AnchorFromPoint](#) property to get the anchor element from the point.

Syntax for MouseDown event, **/NET** version, on:

```csharp
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:

```csharp
private void MouseDownEvent(object sender,
AxEXLABELLib._ILabelEvents_MouseDownEvent e)
{
```

```
}
```

C++
```cpp
void OnMouseDown(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder
```cpp
void __fastcall MouseDown(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi
```delphi
procedure MouseDown(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)
```delphi
procedure MouseDownEvent(sender: System.Object; e:
AxEXLABELLib._ILabelEvents_MouseDownEvent);
begin
end;
```

Powe…
```
begin event MouseDown(integer Button,integer Shift,long X,long Y)
end event MouseDown
```

VB.NET
```vbnet
Private Sub MouseDownEvent(ByVal sender As System.Object, ByVal e As
AxEXLABELLib._ILabelEvents_MouseDownEvent) Handles MouseDownEvent
End Sub
```

VB6
```vb
Private Sub MouseDown(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA
```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(oLabel,Button,Shift,X,Y)
```

RETURN

Syntax for MouseDown event, **/COM** version ^(others), on:

<table>
<tr><td>Java…</td><td><SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT></td></tr>
</table>

<table>
<tr><td>VBSc…</td><td><SCRIPT LANGUAGE="VBScript">
Function MouseDown(Button,Shift,X,Y)
End Function
</SCRIPT></td></tr>
</table>

<table>
<tr><td>Visual Data…</td><td>Procedure OnComMouseDown Short llButton Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
   Forward Send OnComMouseDown llButton llShift llX llY
End_Procedure</td></tr>
</table>

<table>
<tr><td>Visual Objects</td><td>METHOD OCX_MouseDown(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL</td></tr>
</table>

<table>
<tr><td>X++</td><td>void onEvent_MouseDown(int _Button,int _Shift,int _X,int _Y)
{
}</td></tr>
</table>

<table>
<tr><td>XBasic</td><td>function MouseDown as v (Button as N,Shift as N,X as OLE::Exontrol.Label.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Label.1::OLE_YPOS_PIXELS)
end function</td></tr>
</table>

<table>
<tr><td>dBASE</td><td>function nativeObject_MouseDown(Button,Shift,X,Y)
return</td></tr>
</table>

event MouseMove (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user moves the mouse.

Type	Description
Button as Integer	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. Use the AnchorFromPoint property to get the anchor element from the point.

Syntax for MouseMove event, **/NET** version, on:

```csharp
C#
private void MouseMoveEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

```vb
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:

```csharp
C#
private void MouseMoveEvent(object sender,
AxEXLABELLib._ILabelEvents_MouseMoveEvent e)
{
}
```

C++	
| | ```cpp
void OnMouseMove(short Button,short Shift,long X,long Y)
{
}
``` |

C++ Builder	
| | ```cpp
void __fastcall MouseMove(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
``` |

Delphi	
| | ```delphi
procedure MouseMove(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
``` |

Delphi 8 (.NET only)	
| | ```delphi
procedure MouseMoveEvent(sender: System.Object; e:
AxEXLABELLib._ILabelEvents_MouseMoveEvent);
begin
end;
``` |

Powe…	
| | ```
begin event MouseMove(integer Button,integer Shift,long X,long Y)
end event MouseMove
``` |

VB.NET	
| | ```vbnet
Private Sub MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXLABELLib._ILabelEvents_MouseMoveEvent) Handles MouseMoveEvent
End Sub
``` |

VB6	
| | ```vb
Private Sub MouseMove(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
``` |

VBA	
| | ```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(oLabel,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.Label.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Label.1::OLE_YPOS_PIXELS)`
`end function` |

| dBASE | `function nativeObject_MouseMove(Button,Shift,X,Y)`
`return` |

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

```
Private Sub Label1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With Label1
        .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 AxLabel1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXLABELLib._ILabelEvents_MouseMoveEvent) Handles AxLabel1.MouseMoveEvent
    With AxLabel1
        .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 axLabel1_MouseMoveEvent(object sender,
AxEXLABELLib._ILabelEvents_MouseMoveEvent e)
{
    axLabel1.ShowToolTip(axLabel1.get_AnchorFromPoint(-1, -1));
}
```

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

```
void OnMouseMoveLabel1(short Button, short Shift, long X, long Y)
{
    COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
    m_label.ShowToolTip( m_label.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 .Label1
```

```
        .ShowToolTip(.AnchorFromPoint(-1, -1))
    EndWith
endwith
```

event MouseUp (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user releases a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the AnchorFromPoint property to get the anchor element from the point.

Syntax for MouseUp event, **/NET** version, on:

```csharp
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:

```csharp
private void MouseUpEvent(object sender,
AxEXLABELLib._ILabelEvents_MouseUpEvent e)
{
```

```
}
```

C++
```cpp
void OnMouseUp(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder
```cpp
void __fastcall MouseUp(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi
```delphi
procedure MouseUp(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)
```delphi
procedure MouseUpEvent(sender: System.Object; e:
AxEXLABELLib._ILabelEvents_MouseUpEvent);
begin
end;
```

Powe…
```
begin event MouseUp(integer Button,integer Shift,long X,long Y)
end event MouseUp
```

VB.NET
```vbnet
Private Sub MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXLABELLib._ILabelEvents_MouseUpEvent) Handles MouseUpEvent
End Sub
```

VB6
```vb
Private Sub MouseUp(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA
```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(oLabel,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.Label.1::OLE_XPOS_PIXELS,Y as`
`OLE::Exontrol.Label.1::OLE_YPOS_PIXELS)`
`end function` |

| dBASE | `function nativeObject_MouseUp(Button,Shift,X,Y)`
`return` |