ExButton

The ExButton component defines a command button. A command button on a form can start an action or a set of actions. For example, you could create a command button that opens another form. The control provides predefined button skins for Windows XP, Windows 95/98, and Mac 8.x buttons. Create your own skins for your buttons in minutes, using a WYSYWG skin builder. The ability to specify everything that control needs like graphical objects, transparent skins, HTML captions as simple text makes the exButton one of the best. The exButton control easily replaces the Standard Windows button by supporting most of the same properties, methods and events. In addition, you have complete control over how the button is displayed. The skin method, in it's simplest form, uses a single graphic file assigned to the client area of the button. 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 button.

Features include:

- **WYSWYG Skin Builder** (ability to define your button faces in minutes)
- **WYSWYG Template/Layout Editor** support
- **Unlimited** options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the object's background
- Horizontal, Vertical, Rotate, Mirror support
- Rectangular or **Round/Circular** button support
- Predefined skins for **Windows XP, Windows 95/98, and Mac 8.x** buttons, Windows XP not required
- Ability to specify **HTML** multiple lines caption.
- Multiple-Lines HTML Tooltip support
- Transparent color support
- Focusable support
- Ability to specify the control's background color without changing the visual appearance
- Ability to build toolbars with your own style
- Ability to assign different skins for different button's states
- Ability to assign icons, images, **transparent images** to the control using **BASE64 encoded** strings
- **Text Decorations** support, like gradient, outlined characters, shadow, and so on
- Ability to align the caption or the image anywhere on the button's client area
- Ability to specify a picture on the button's background
- Ability to resize the button without losing the corners
- Normal, Push-Like, Check Box, Custom modes support
- Hot, Focus state support

How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

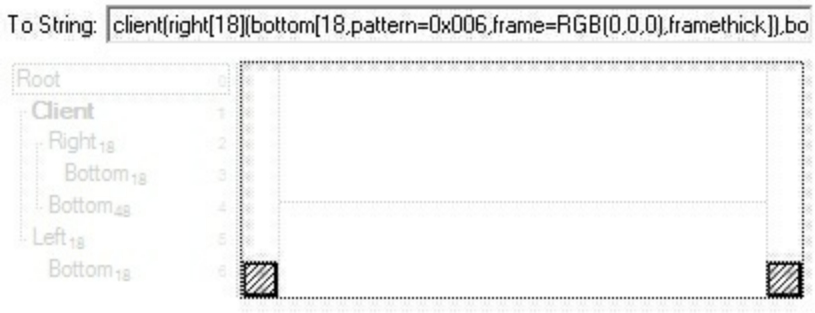
constants AlignmentEnum

Specifies the image or caption alignment. Use the Alignment property to change the horizontal alignment for the caption.Use the ImageAlignment property to change the horizontal alignment for the image. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area

Name	Value	Description
exLeft	0	The object is left aligned.
exCenter	1	The object is centered.
exRight	2	The object is right aligned.

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

FormatAnchor property customizes the visual effect for anchor elements.

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAA " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAA string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

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

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

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

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

exTextExt 4

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

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

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

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

gradient-center

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

outlined

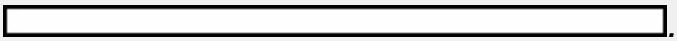
- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2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

shadow

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

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

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

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

constants 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.

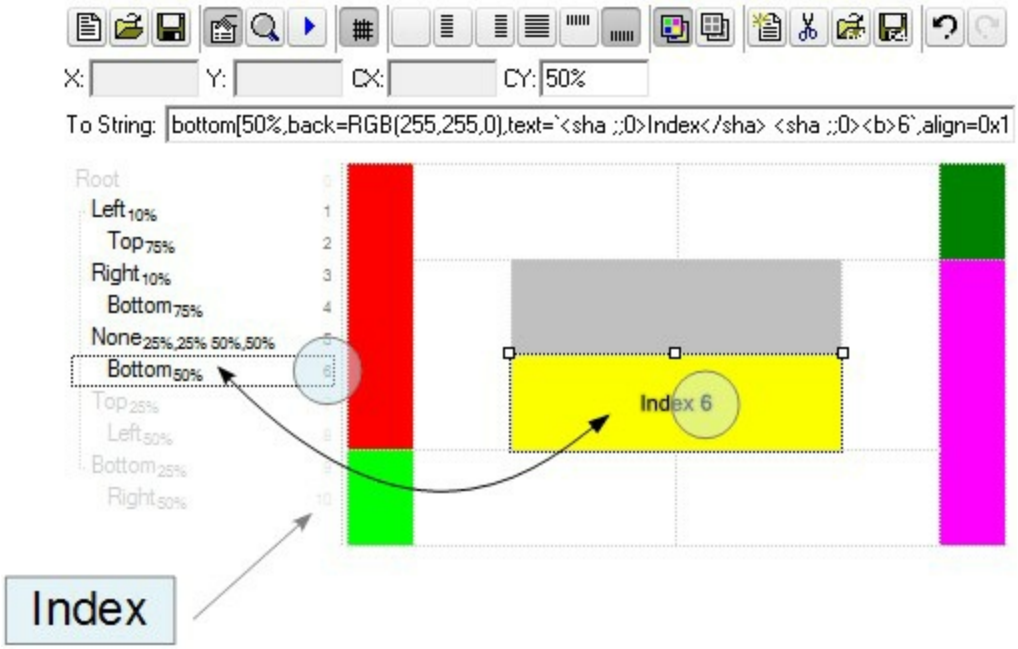
constants ModeEnum

Describes how the button's State is changed when user clicks the button. Use the Mode property to change the control's mode. By default, the control's mode is exButton. The control fires the Click event when user clicks the button. The button's State property is changed depending on the control's Mode property.

Name	Value	Description
exButton	0	The control acts like a normal button.
exPushLike	1	The control toggles the states when user clicks the control.
exCustomMode	2	The user is free to choose what state is next when user clicks the control.

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:*

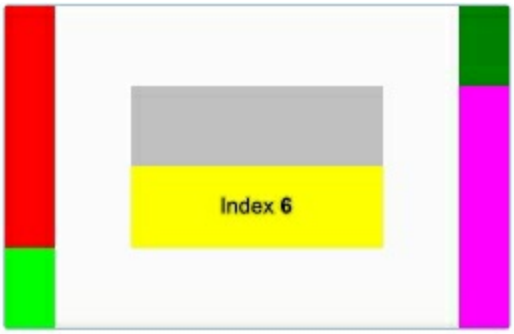


In this sample, there are 11 objects that composes the EBN, so the Index property goes from 0 which indicates the root, and 10, which is the last item in the list

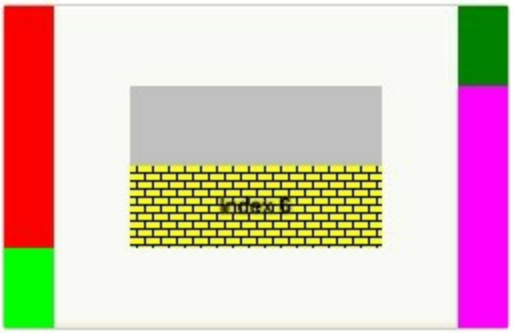
So, let's apply this format to an object, to change the exPatternExt property for the object with the Index 6:

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 a picture object is displayed. Use the Picture property to load a picture on the control's background. Use the PictureDisplay property to specify how the Picture is displayed on the control's background.

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 SideEnum

A SideEnum expression that indicates the side being modified. Use the IncClientState property to adjust the button's client area.

Name	Value	Description
exLeftSide	0	The left side to be modified.
exTopSide	1	The top side to be modified.
exRightSide	2	The right side to be modified.
exBottomSide	3	The bottom side to be modified.

constants StateEnum

Describes the control's state. Use the State property to determine the button's state. Use the Skin property to assign a skin for a specified state. Use the FocusSkin property to assign a skin for a specified state when control has the focus. Use the Mode property to specify the mode how the button is running. The control supports the following states:

Name	Value	Description
exNormal	0	The button is not pressed.
exPushed	1	The button is pressed.
exHot	2	The cursor is over the button.
exDisabled	3	The button is disabled. Use the Enabled property to enable or disable the button.
exCustom	4	Custom state.

constants StyleEnum

The StyleEnum expression defines a predefined visual appearance for your button. Use the Style, Skin, FocusSkin property to change the button's visual appearance. The following predefined styles are supported:

Name	Value	Description
exCustom4	-4	Custom skin
exCustom3	-3	Custom skin
exCustom2	-2	Custom skin
exCustom1	-1	Custom skin
exDefault	0	
exMAC	1	 The button will look and act like the Mac 8.x OS button style.
exXPBlue	2	 The button will look and act like the Windows XP buttons blue color scheme.
exXPGreen	3	 The button will look and act like the Windows XP buttons olive green color scheme.
exXPSilver	4	 The button will look and act like the Windows XP buttons silver color scheme.

constants VAlignmentEnum

Specifies the image or caption alignment. Use the VAlignment property to change the vertical alignment for the caption. Use the ImageVAlignment property to change the vertical alignment for the image. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area

Name	Value	Description
exTop	0	The object is top aligned.
exMiddle	1	The object is centered.
exBottom	2	The object is bottom aligned.

Button 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: {F3A2203A-6B28-4A74-9DC9-4065D1C0A29D}. The object's program identifier is: "Exontrol.Button". The /COM object module is: "ExButton.dll"

The exButton control is designed to enhance your WindowsŽ-based programs by offering the look-and-feel of present GUI design elements. The control provides predefined button skins for Windows XP, Windows 95/98, and Mac 8.x buttons. Create your own skins for your buttons in minutes, using a WYSYWG skin builder. The ability to specify everything that control needs like graphical objects, transparent skins, HTML caption as simple text makes the exButton one of the most wanted button control on the market. The exButton control easily replaces the Standard Windows button by supporting most of the same properties, methods and events. How to replace my old VB buttons with this new button?

The Exontrol's Button object supports the following properties and methods:

Name	Description
Alignment	Aligns the caption in the control.
AllowHotState	Specifies whether the control displays the hot state when the cursor is over the control.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
BackColor	Specifies the control's background color.
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 do changes 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 button's caption.
Debug	Specifies whether the control displays debugging information.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
Focusable	Gets or sets a value that indicates whether the control can receive focus.
FocusSkin	Specifies the skin file to display the specified state, when control has the focus.
FocusSkinV	Specifies the skin file to display the specified state, when control has the focus.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
ForeColorState	Specifies the foreground color for a specified state.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
HFit	Specifies a value that indicates the horizontal offset to fit image with the caption.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Image	Specifies the image being displayed.
ImageAlignment	Specifies the image's alignment.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Image List Control.
ImageSize	Retrieves or sets the size of icons the control displays.
ImageVAlignment	Specifies the image's vertical alignment.
IncClientState	Adjusts the control's client area for the specified side and state.
Mode	Specifies the control's mode.
MouseIcon	Sets or returns a value that determines a custom Icon to

	be displayed when the pointer moves over the control.
MousePointer	Sets or returns a value that determines the MousePointer to be displayed when the pointer moves over the control.
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.
RenderType	Specifies the way colored EBN objects are displayed on the component.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
Rotate	Rotates the HTML caption.
ShowFocusRect	Sets or returns a value that determines whether or not the focus rectangle should be shown.
Skin	Specifies the skin file to display the specified state.
SkinV	Specifies the skin file to display the specified state.
State	Specifies the control's state.
Style	Specifies the control's style.
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.
UseFocusSkin	Specifies whether the focus skins are used when control has the focus.

UserData	Gets or sets the user-definable data for the current object.
UseTransparency	Specifies whether the control supports transparency. The transparent regions in the control's skin indicates the transparency.
VAlignment	Specifies the caption's vertical alignment.
Version	Retrieves the control's version.
VFit	Specifies a value that indicates the vertical offset to fit image with the caption.
WordWrap	Indicates whether a multiline text automatically wraps words to the beginning of the next line when necessary.

property Button.Alignment as AlignmentEnum

Aligns the caption in the control.

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

Use the Caption property to assign a caption to the button. By default, the Alignment property is exCenter. Use the Alignment property to align the caption n the button. Use the VAlignment property to vertically align the caption in the button. Use the ImageAlignment property to align the the image of the button. Use the ImageVAlignment property to vertically align the image in the button. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area.

property Button.AllowHotState as Boolean

Specifies whether the control displays the hot state when the cursor is over the control.

Type	Description
Boolean	A boolean expression that indicates whether the exHot state is displayed when the cursor is over the button.

By default, the AllowHotState property is false. Use the Style property to specify the control's style. The Style property changes the AllowHotState and UseFocusSkin properties based on the style used. Use the Skin property to assign custom skins for specified states. If the AllowHotState property is True, the Skin(exHot) gets displayed when the cursor is over the button.

Please be aware that Style property changes the following properties, based on the style chosen:

- AllowHotState property
- UseFocusSkin property
- Skin
- FocusSkin

If you require certain value for the AllowHotState property you have to change the AllowHotState property after changing the Style property.

For instance, the exDefault predefined style for hot state (when the cursor is over the button).

property Button.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;tooltip> anchor elements to add hyperlinks to cell's caption. The tooltip is shown when the cursor hovers an anchor element with a tooltip parameter. The control fires the AnchorClick event when the user clicks an anchor element. The MouseMove event is generated continually as the mouse pointer moves across the control.

method Button.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 Button1_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.

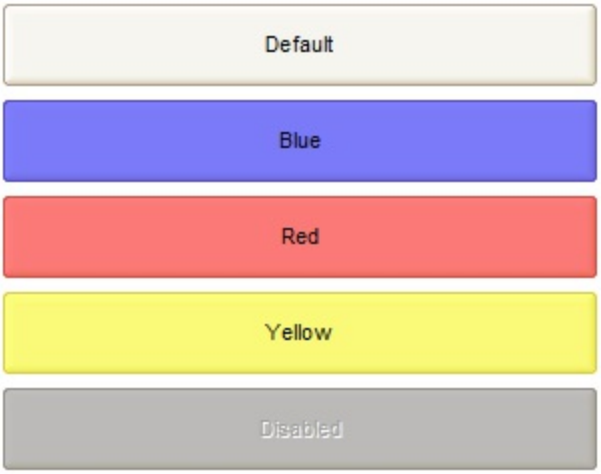
property Button.BackColor as Color

Specifies the control's background color.

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

The BackColor property changes the control's background color. The background color is applied to the current style/visual appearance. The Style property specifies the control's visual appearance. Use the Skin/SkinV method to customize the current visual appearance using EBN objects. The color for the button's visual appearance is not changed if the BackColor property is white or black. The Enabled property specifies whether the control is enabled or disabled. The ForeColorState(exDisabled) specifies the color for the disabled state. Using the BackgroundExt property you have unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the object's background.

The following screen shot shows the button's look once the BackColor property is changed and Style property is exDefault:



The following screen shot shows the button's look once the BackColor property is changed and Style property is exMAC:

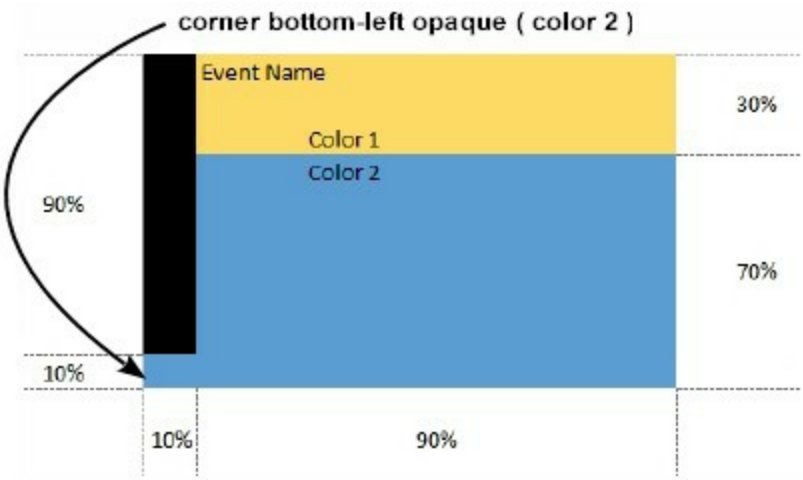
property Button.BackgroundExt as String

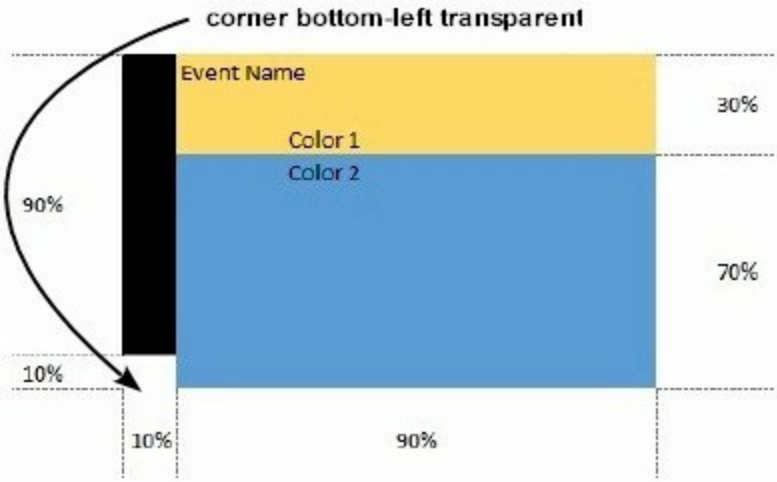
Indicates additional colors, text, images that can be displayed on the object's background using the EBN string format.

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

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

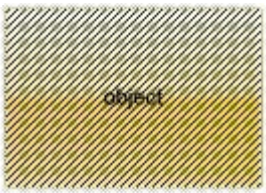
Complex samples:



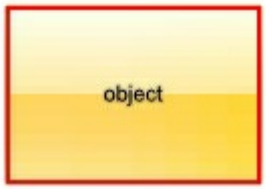


Easy samples:

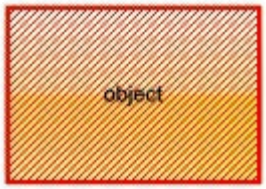
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



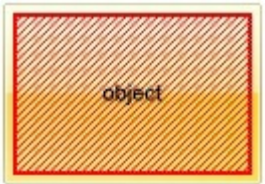
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



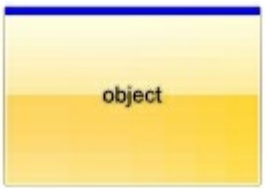
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



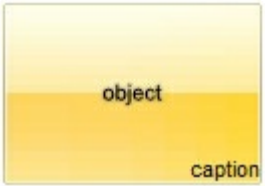
- "[[patterncolor=RGB(255,0,0)] (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



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

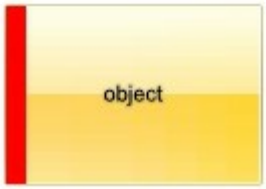


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

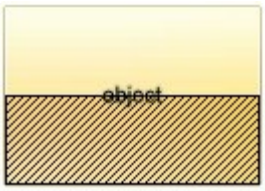


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

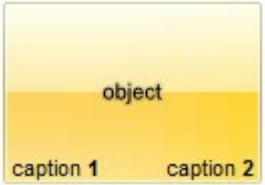
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



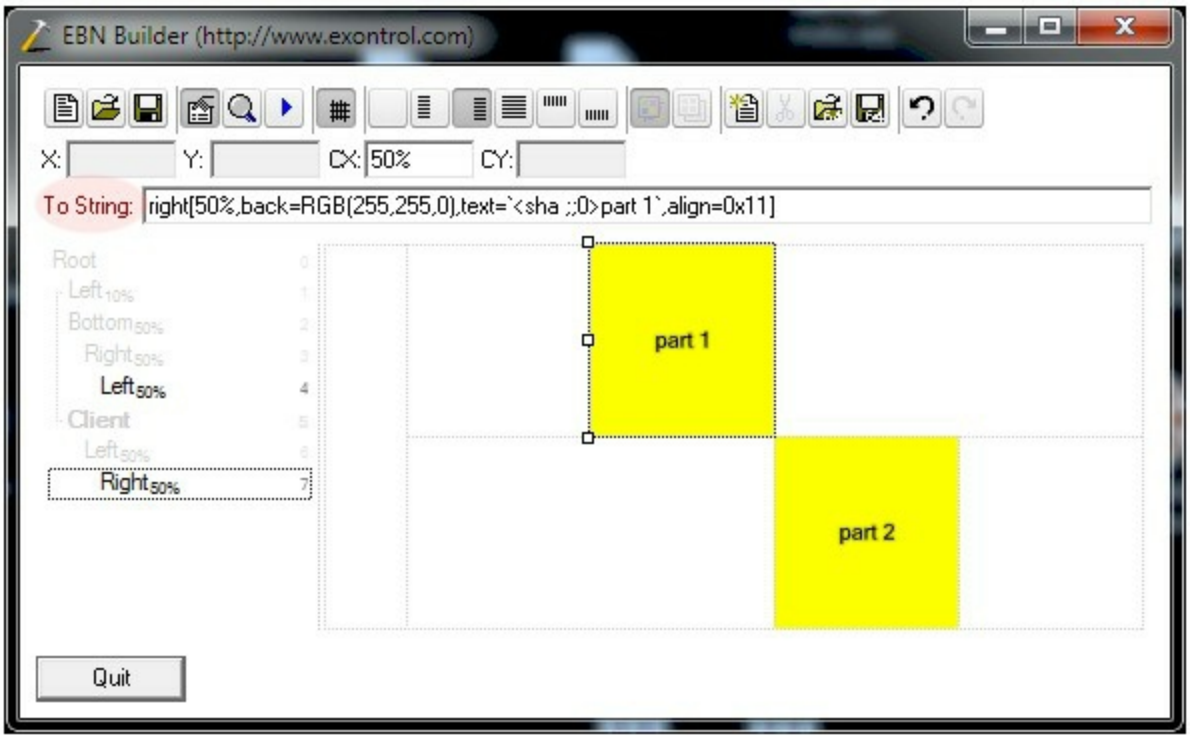
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



- "root[text=`caption <b>2`,align=0x22](client[text=`caption <b>1`,align=0x20])", shows the caption **1** aligned to the bottom-left side, and the caption **2** to the bottom-right side



*The Exontrol's [eXButton](#) WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field as shown in the following screen shot:*



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

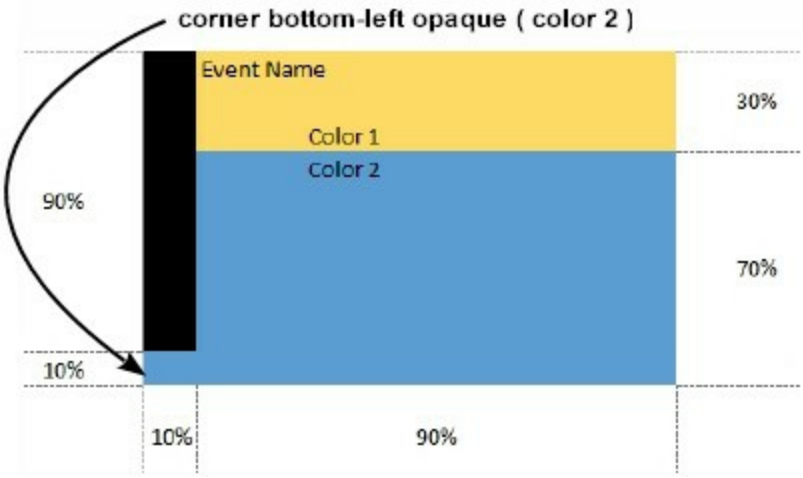
The **EBN String Format** syntax in BNF notation is defined like follows:

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

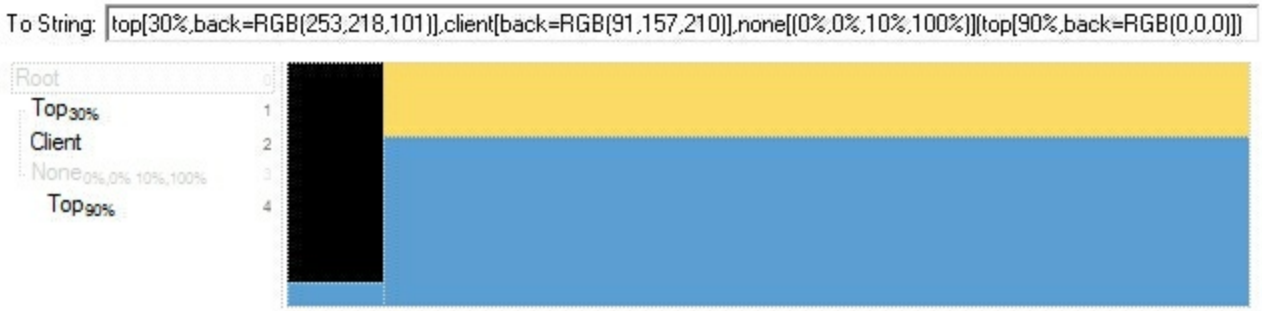
```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"  "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number>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

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

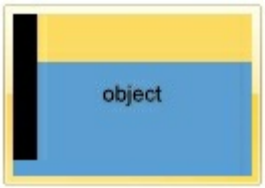
Now, lets say we have the following request to layout the colors on the objects:



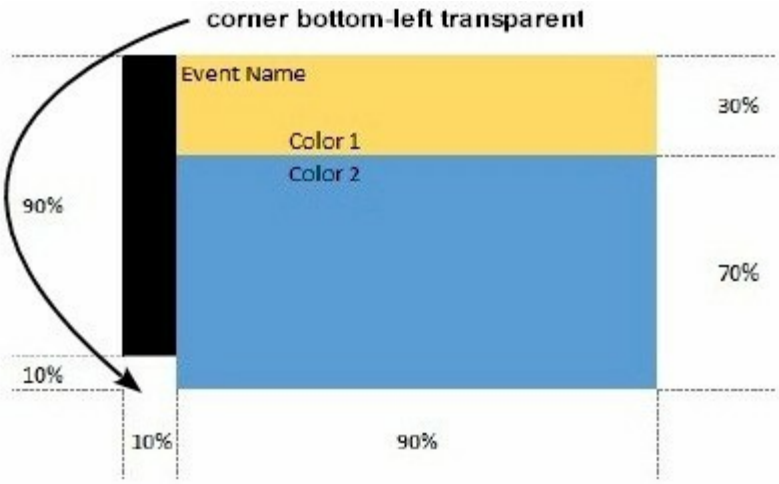
We define the BackgroundExt property such as "top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100
(top[90%,back=RGB(0,0,0)])", and it looks as:



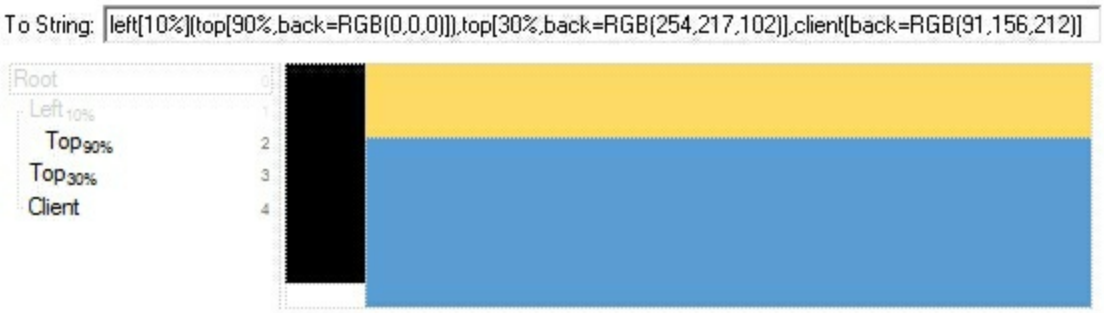
so, if we apply to our object we got:



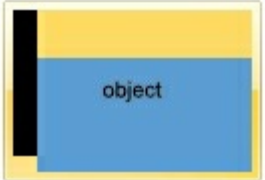
Now, lets say we have the following request to layout the colors on the objects:



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

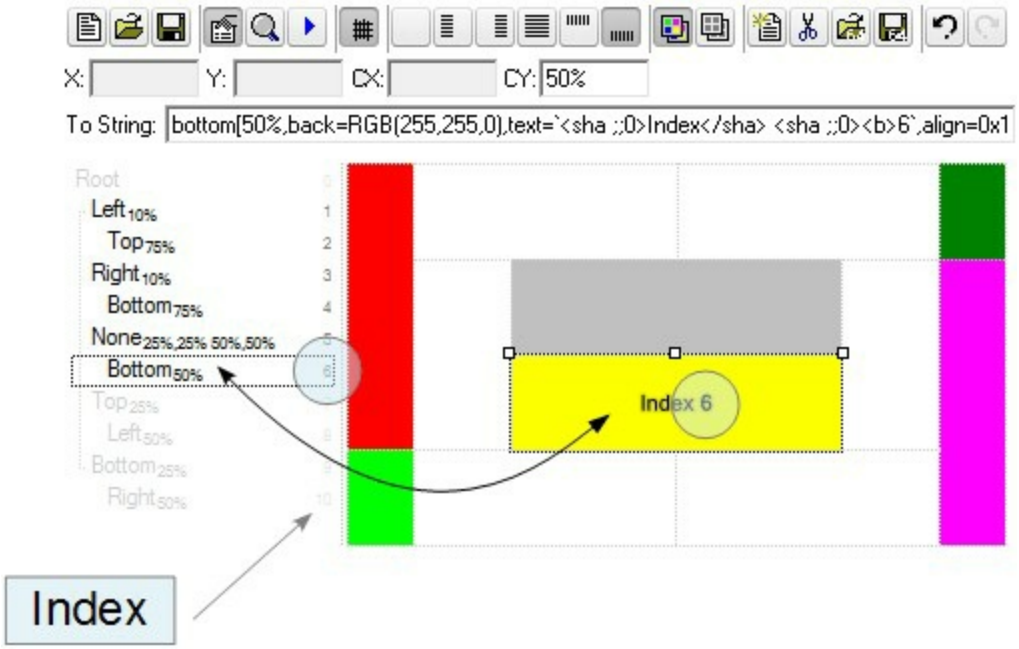


so, if we apply to our object we got:


object

property Button.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.

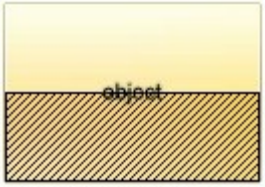
You can access/define/change the following UI properties of the element:

- **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*

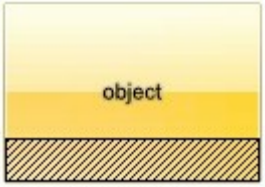
expression)

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 Button.BeginUpdate ()

Maintains performance when do changes one at a time. This method prevents the control from painting until the EndUpdate method is called.

Type	Description

Use the BeginUpdate and EndUpdate method when multiple changes are required at the same time.

The following sample locks painting the button's appearance while changes is performed:

```
With Button1

  .BeginUpdate
    .BorderHeight = 20
    .BorderWidth = 20

    .Picture = LoadPicture("C:\WINNT\Web\tips.gif")
    .PictureDisplay = Tile

    Dim s As String
    s =
"gBHJJGHA5MJAAEIe4AAAFh0OCERiQbigwEobAsXCAljkcHYwDYQkAli0iGAwHYlCA7HZQIp

    s = s +
"mm0IpYjEUogW2JJtCwVBpFIJQKhWYQdAKUwSAwCYGAqSw+DmBRrFYS4YH+AhNiSGYM

    s = s +
"SWBzh4CQMEIQyxSiYAMOILwfBMDSDEAt/DXB0iIFYJAQoXwlBIFqBIZYqhWDGHkAgM4tRa

    s = s +
"YdoNwVAtECHQZ4kxUDrGeAIHwTRaiMB0AsJIxQVguDeCoWQ9B3i/BYJALA1B/gGBwIgeAD

    s = s +
"cBAAQNQ2BBAeIcFAjgLC+EoAAO4mRzA+DGIsVIAgMBwBABNAJAwAAB1BGBBAOgkAIgC

    .Caption = "&Help;"
```

```
        .Image = s
        .VFit = 6

    .EndUpdate
End With
```

property Button.BorderHeight as Long

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

Type	Description
Long	A long expression that indicates the height of the control's margin, in pixels.

The BorderHeight and BorderWidth properties determines the area where the skin is displayed. By default, the BorderHeight property is 0If the BorderWidth and BorderHeight properties are 0(zero), the skin uses the entire control's client area. In this case we can say that the control has no margins. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area. The BackColor and Picture properties have effect on the control's margins and on the transparent areas of the skin.

property Button.BorderWidth as Long

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

Type	Description
Long	A long expression that indicates the width of the control's margins in pixels.

The BorderHeight and BorderWidth properties determines the area where the skin is displayed. By default, the BorderWidth property is 0. If the BorderWidth and BorderHeight properties are 0(zero), the skin uses the entire control's client area. In this case we can say that the control has no margins. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area. The BackColor and Picture properties have effect on the control's margins and on the transparent areas of the skin.

property Button.Caption as String

Specifies the button's caption.

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

The button allows for an access key to be set in the caption by placing an ampersand (&) before the character to be used for the access key. The button's first ampersand specifies the access key. Use the **<img>** HTML tag to insert icons inside the button's caption

The Caption supports the following HTML tags:

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

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

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

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

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

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a

known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

The WordWrap property to indicates whether a multi-lines text automatically wraps words to the beginning of the next line when necessary. Use the Image property to assign an image to the button. Use the Picture property to assign a picture on the control's background. Use the Alignment property to align the caption in the button. Use the VAlignment property to vertically align the caption in the button. Use the Style or Skin property to change the button's visual appearance.

All of the properties and methods for all objects in the Exontrol's exButton can be assigned at design time using the WYSWYG Template /Layout editor. Open the control in design mode, and select Properties from its context menu. The Template language uses the x-script language (very simple), that can be used to initializes the properties and methods at design time.

The following sample initializes the control's image and caption at design mode, using the Template page:

```
' Specifies the control's caption
Caption = "Exontrol's exButton component is our answer to your GUI needs"
WordWrap = True

' https://www.exontrol.com/sg.jsp?content=support/faq#eximages
Image =
"gBHJJGHA5MIwAEIe4AAAFhwbiAliQwig7ixFjBQjRbjhljxwkB7kSFkiQkyblCllSwli7lzFmDQm1
```

The button's visual appearance in this case will be:

property Button.Debug as Boolean

Specifies whether the control displays debugging information.

Type	Description
Boolean	A boolean expression that specifies whether the control displays debugging information.

Only for internal use.

property Button.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 button is enabled. Use the Enabled property to disable the control. When control is disabled, the control displays the exDisabled skin. Use the ForeColorState property to assign a different foreground color when the control is disabled. By default, the ForeColorState(exDisabled) is COLOR_GRAYTEXT system color.

method Button.EndUpdate ()

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

Type	Description

Use the BeginUpdate and EndUpdate method when multiple changes are required at the same time.

The following sample locks painting the button's appearance while changes is performed:

```
With Button1

   .BeginUpdate
      .BorderHeight = 20
      .BorderWidth = 20

      .Picture = LoadPicture("C:\WINNT\Web\tips.gif")
      .PictureDisplay = Tile

      Dim s As String
      s =
"gBHJJGHA5MJAAEIe4AAAFh0OCERiQbigwEobAsXCAljkcHYwDYQkAli0iGAwHYlCA7HZQIp

      s = s +
"mm0IpYjEUogW2JJtCwVBpFIJQKhWYQdAKUwSAwCYGAqSw+DmBRrFYS4YH+AhNiSGYM

      s = s +
"SWBzh4CQMEIQyxSiYAMOILwfBMDSDEAt/DXB0ilFYJAQoXwlBIFqBIZYqhWDGHkAgM4tRa

      s = s +
"YdoNwVAtECHQZ4kxUDrGeAIHwTRaiMB0AsJIxQVguDeCoWQ9B3i/BYJALA1B/gGBwIgeAD

      s = s +
"cBAAQNQ2BBAeIcFAjgLC+EoAAO4mRzA+DGIsVIAgMBwBABNAJAwAAB1BGBBAOgkAIgC



      .Caption = "&Help;"
      .Image = s
```

```
        .VFit = 6

    .EndUpdate
End With
```

property Button.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 Button.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 String 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 the control's background color:

```
Debug.Print G2antt1.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 Button.Focusable as Boolean

Gets or sets a value that indicates whether the control can receive focus.

Type	Description
Boolean	Gets or sets a value that indicates whether the control can receive focus.

By default, the Focusable property is True. In other words, the control receives the focus once the user clicks the control. Use the Focusable property to prevent receiving the focus, when user clicks the button. You can use this property to simulate buttons in a toolbar control. The Focusable property does not specify whether the control gains or loses the focus (changing the focus). This property has no effect if the user navigate the controls in the form using the keyboard. The control gains the focus using the keyboard (TAB or SHIFT+TAB) if the TabStop property of the control is True (by default). In conclusion, set the Focusable and TabStop property on False, to prevent receiving the focus using the mouse and the keyboard.

method Button.FocusSkin (State as StateEnum, File as String)

Specifies the skin file to display the specified state, when control has the focus.

Type	Description
State as StateEnum	A StateEnum expression that indicates the state skin being changed.
File as String	A String expression that indicates the BASE64 encoded string that holds a skin file (*.ebn), or a path to the skin file (*.ebn). The skin file must be created using the Exontrol's ExButton Builder. Use the Exontrol's exImages tool to build BASE 64 encoded strings on the skin file you have created. If the File parameter is an empty string the skin is erased, so your button displays only the Image and the Caption of the button without a visual appearance (skin).

The FocusSkin method assign a new skin to the specified state. The focused skin is displayed ONLY if the UseFocusSkin property is True, and the control has the focus. By default, the exMAX style contains a focused skin, and changes the UseFocusSkin property on True. Use the Skin method to assign different skins on specified states when the control is not focused or when the UseFocusSkin property is False. Use the FocusSkin and UseFocusSkin properties to assign a different visual appearance to the button when it has the focus. Use the ShowFocusRect property to display a thin rectangle around the button's caption or button's image when the button has the focus. The Style property deletes any previous skin used. Call the FocusSkin property after Style, or Skin property if you require a different focused skin. Use the ForeColorState property to assign a different foreground color for certain state. Use the FocusSkinV method to load EBNs from resources.

Please be aware that Style property changes the following properties, based on the style chosen:

- AllowHotState property
- UseFocusSkin property
- Skin
- FocusSkin

We would recommend taking a look over the following articles:

- How to build my own skin file?
- How do I assign a skin file to my button?

method Button.FocusSkinV (State as StateEnum, Skin as Variant)

Specifies the skin file to display the specified state, when control has the focus.

Type	Description
State as StateEnum	A StateEnum expression that indicates the state skin being changed.
Skin as Variant	A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file or a String expression that indicates the BASE64 encoded string that holds a skin file (*.ebn), or a path to the skin file (*.ebn). The skin file must be created using the Exontrol's ExButton Builder. Use the Exontrol's exImages tool to build BASE 64 encoded strings on the skin file you have created. If the File parameter is an empty string the skin is erased, so your button displays only the Image and the Caption of the button without a visual appearance (skin).

The FocusSkinV is similar with FocusSkin method, excepts that it can loads the EBN files from safe arrays or in other words from resources. The button provides multiple states like: normal, pushed, hot, disabled, custom, focused and so on. Each state has an associated skin that's displayed when certain state occurs. The UseTransparency property specifies whether the control supports transparency. The transparent regions in the control's skin indicates the transparency of the button.

There are several options to provide EBN files in your project as follows:

- (**path**) The path to the EBN file. This option is useful when your application installs files on the client's machine so you can provide the path to EBN files.

```
With Button1
   .FocusSkinV exNormal, "C:\Program Files\Exontrol\EBN\vistasel.ebn"
End With
```

- (**string**) The BASE64 encoded string that holds the EBN file. This option is useful if you provide EBN objects in the control's Template page or in code. The Exontrol's exImages tool generates BASE64 encoded strings from EBN files.

```
With Button1
   Dim s As String
   s =
"gBFLBCJwBAEHhEJAEGg4BHQDg6AADACAxRDAMgBQKAAzQFAYahyGCGAA
```

```
  s = s +
"SiOKRKEaFYkmiWYwmuIRliOLhBDcKZ6gSl4qDqCokimahqiaJYqk2SYwmyJwg

  .FocusSkinV exNormal, s
End With
```

In order to generate the BASE64 encoded string from your EBN file do the following:

- Run the eXImages tool
- Run the Windows Explorer and select or locate the EBN file. Press the CTRL + C or drop the EBN file in the middle panel (Drag here files such of .bmp, .gif, .ebn, ...)
- The clipboard contains the generated BASE64 string, or you can copy it from the right panel of the eXImages tool. Generally, the string is long, so you can use the s definition to insert it to your code.

- (**array**) A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. This option is useful if you want to provide the EBN files in the project resources. The idea is that you have to provide a safe array of bytes to the Skin parameter of the FocusSkinV method. For instance, the VB6 provides the LoadResData function, the VB/NET or C# provides an internal class Resources where all items in the resources can be accessed through public properties.

  VB6

  ```
  With Button1
    .FocusSkinV exNormal, LoadResData(101, "CUSTOM")
  End With
  ```

  In order to insert the EBN file to the project resources do the following:

  - Click the VB Resource Editor button in the toolbox.
  - Once the VB Resource Editor tool is opened, click the Add Custom Resource ... button in the toolbox
  - Locate, Select the EBN file in the opened file/folder dialog, and press Open button

- The "CUSTOM"\101 item should be inserted in the resource file.
- Click the Save button, so the RES file is being associated with your project.

VB/NET

```
With Exbutton1
   .FocusSkinV(exNormal, WindowsApplication1.My.Resources.vistasel)
End With
```

In order to insert the EBN file to the project resources do the following:

- Select the Project\Properties... from the VS menu
- Click the Resources page
- Click the Add Resource and then Add Existing File...
- Locate, Select the EBN file in the opened file/folder dialog, and press Open button
- The vistasel item is being generated and so it can be accessed in code using: WindowsApplication1.My.Resources.vistasel

C#

```
exbutton1.FocusSkinV(exNormal,
WindowsApplication1.Properties.Resources.vistasel);
```

In order to insert the EBN file to the project resources do the following:

- Select the Project\Properties... from the VS menu
- Click the Resources page
- Click the Add Resource and then Add Existing File...
- Locate, Select the EBN file in the opened file/folder dialog, and press Open button
- The vistasel item is being generated and so it can be accessed in code using: WindowsApplication1.Properties.Resources.vistasel

property Button.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object being used.

Use the Font property to assign a new font to the control. Use the HTML tags like <b> to bold parts of control's caption. Use the Caption property to assign a caption to the button. The font must be of type StdFont in the OLE Automation type library. When the control is initially placed on a form or other container object, the Font property will automatically be set to the "Arial" font.

property Button.ForeColor as Color

Specifies the control's foreground color.

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

Use the ForeColor property to assign a foreground color to your button. Use the HTML tags like <fgcolor>, <bgcolor> in the Caption property to colorize parts of the caption. Use the ForeColorState property to assign a color for a given state. Use the ForeColorState property to assign a foreground color when control is disabled.

property Button.ForeColorState(State as StateEnum) as Color

Specifies the foreground color for a specified state.

Type	Description
State as StateEnum	A StateEnum expression that indicates the foreground's color being changed.
Color	A Color expression being used when the control's State property is State.

Use the ForeColorState property to assign foreground colors for states of the button. The ForeColor and ForeColorState(exNormal) are equivalents.

property Button.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 Button.HFit as Long

Specifies a value that indicates the horizontal offset to fit image with the caption.

Type	Description
Long	A long expression that indicates the the horizontal offset to fit image with the caption.

By default, the HFit property is 0(zero). Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area.

property Button.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 Caption property to specify the caption of the button, using HTML format.

property Button.hWnd as Long

Retrieves the control's window handle.

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

This property is useful for executing certain API calls. The property is read only at runt time.

property Button.Image as Variant

Specifies the image being displayed.

Type	Description
Variant	A long expression that indicates the index of icon being displayed, a string expression that specifies the path to a picture file, a Picture object being displayed, a string expression that indicates the BASE64 encoded string that holds a picture/image object. Use the eximages tool to save your picture as base64 encoded format., or a IPictureDisp object that holds the picture being displayed on the button.

Use the Image property to assign a picture to your button. Use the Images property to add icons to the control. If you have multiple icons added to the Images collection you can use long expression to change the index of icon being displayed. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ImageAlignment property to align the image in the button. Use the ImageVAlignment property to align vertically the image in the button. Use the Caption property to assign a caption to the button. Use the Style or Skin property to change the visual appearance for your button. Use the **<img>** HTML tag to insert icons inside the button's caption.

The following sample loads the picture from a file:

```
With Button1
   .Image = "D:\temp\icons\settings.gif"
End With
```

The following sample assigns the first icon from an ImageList control:

```
With Button1
   .Images ImageList1.hImageList
   .Image = 1
End With
```

The following sample assigns a Picture objects:

```
With Button1
   .Image = Picture1.Picture
End With
```

The following sample loads the picture from an BASE64 encoded string (Use the eximages tool to save your picture as base64 encoded format)

```
With Button1
   Dim s As String
   s =
"gBHJJGHA5MIwAEIe4AAAFhwbiAliQwig7ixFjBQjRbjhljxwkB7kSFkiQkyblCllSwli7lzFmDQm1

   s = s +
"AjcCwgAAIJBhQBQkHGL4gDaNBokkZQMiwUAuioJQiCAQYsHMcwwEIeoigAYIogsGIwFKIYI(



   .Image = s
End With
```

The following sample displays the first icon in the Images collection:

```
With Button1
   .Images
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExm

   .Image = 1
End With
```

All of the properties and methods for all objects in the Exontrol's exButton can be assigned at design time using the WYSWYG Template /Layout editor. Open the control in design mode, and select Properties from its context menu. The Template language uses the x-script language (very simple), that can be used to initializes the properties and methods at design time.

The following sample initializes the control's image and caption at design mode, using the Template page:

```
' Specifies the control's caption
Caption = "Exontrol's exButton component is our answer to your GUI needs"
WordWrap = True

' https://www.exontrol.com/sg.jsp?content=support/faq#eximages
```

```
Image =
"gBHJJGHA5MIwAEIe4AAAFhwbiAliQwig7ixFjBQjRbjhljxwkB7kSFkiQkybICllSwli7lzFmDQmT
```

The button's visual appearance in this case will be:

property Button.ImageAlignment as AlignmentEnum

Specifies the image's alignment.

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

Use the Image property to load an image in the button. Use the ImageAlignment property to align the button's image. Use the ImageVAlignment property to vertically align the image in the button. By default, the ImageAlignment property is exCenter. Use the Alignment property to align the caption in the control. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area

method Button.Images (Handle as Variant)

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

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

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

Use the Images method to load a list of icons to the button. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the Image property to assign a picture/image to the button.

The following sample adds two icons to the control's images list and change the button's icon when user clicks the button:

```
Private Sub Button1_Click()
With Button1
   .BeginUpdate
      .Image = (.Image) Mod 2 + 1
   .EndUpdate
End With
End Sub

Private Sub Form_Load()
With Button1
   .BeginUpdate
      .Mode = exPushLike
      .Images
"gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8pIUrlktl0vmExn

      .Images
"gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8pIUrlktl0vmExn

      .Image = 1
   .EndUpdate
End With
End Sub
```

property Button.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 Button.ImageVAlignment as VAlignmentEnum

Specifies the image's vertical alignment.

Type	Description
VAlignmentEnum	A VAlignmentEnum expression that indicates the vertical alignment for the button's image.

Use the Image property to assign an image to the button. By default, the ImageVAlignment property is exMiddle. Use the ImageVAlignment property to change the vertical alignment for the button's image. Use the ImageAlignment property to change the horizontal alignment for the image. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area

property Button.IncClientState(State as StateEnum, Side as SideEnum) as Long

Adjusts the control's client area for the specified side and state.

Type	Description
State as StateEnum	A StateEnum expression that indicates the State's client area being changed.
Side as SideEnum	A SideEnum expression that indicates the side being changed.
Long	A long expression that indicates the offset used to indent the button's client area, in pixels.

The Caption and the Image property are displayed on the button's client area. Each skin defines an area called client area. On the client area of the skin you can get displayed the 'Caption' string. The control's client area and the button's client area is different. The control's client area is determined by the control's window, since the button's client area is determined by the properties BorderWidth, BorderHeight, and IncClientState. Use the IncClientState(exPushed) property to indent the area where the Image and the Caption of the button are displayed. By default the IncClientState(exPushed, exLeftSide) and IncClientState(exPushed, exLeftSide) are 2. Set IncClientState(exPushed, exLeftSide) and IncClientState(exPushed, exTopSide) on 0 (zero) to avoid indenting the area where the Image and the Caption of the button are displayed. The Style property changes the IncClientState property based on each style.

property Button.Mode as ModeEnum

Specifies the control's mode.

Type	Description
ModeEnum	A ModeEnum expression that indicates how the button's state is changed when the user clicks the button.

By default, the control's Mode property is exButton. The State property is changed by the control depending on the control's Mode property. Use the Style or Skin property to change the visual appearance for the button. The control fires the Click event when user clicks the button. The control fires also the Click event if the button has the focus, and the user releases the SPACE key. Use the KeyDown event to avoid firing the Click event when user presses the SPACE key like in the following sample:

```
Private Sub Button1_KeyDown(KeyCode As Integer, Shift As Integer)
   KeyCode = 0
End Sub
```

property Button.MouseIcon as IPictureDisp

Sets or returns a value that determines a custom Icon to be displayed when the pointer moves over the control.

Type	Description
IPictureDisp	A reference to an Icon object. Can be a reference to another objects Picture property or loaded using the LoadPicture method, but the graphic must be of type Icon.

The MouseIcon property is ignored until the MousePointer value is set to 99

The following VB sample shows how you can display a cursor using an ICO file (32x32)

```
With Button1
   .MouseIcon = LoadPicture(App.Path + "\cursor.ico")
   .MousePointer = 99
End With
```

property Button.MousePointer as Long

Sets or returns a value that determines the MousePointer to be displayed when the pointer moves over the control.

Type	Description
Long	A long expression that indicates the MousePointer to be displayed, as described below.

When MousePointer is set to exCustom, the control will use the icon that is set in the MouseIcon property. The valid values for the MousePointer property are:

0 - exDefault Default MousePointer
1 - exArrow Arrow
2 - exCrossHair Crosshair
3 - exIBeam Ibeam
4 - exIconPointer Icon
5 - exSizePointer Size
6 - exSizeNESW Size Northeast / Southwest
7 - exSizeNW Size North / West
8 - exSizeNWSE Size Northwest / Southeast
9 - exSizeWE Size West / East
10 - exUpArrow Up arrow
11 - exHourglass Hourglass
12 - exNoDrop "No" symbol

13 - exArrowHourglass Arrow with hourglass
14 - exArrowQuestion Arrow with question mark
15 - exSizeAll Size all
16 - exPointer Hand

99 - exCustom Custom pointer, set in MouseIcon property

property Button.Picture as IPictureDisp

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

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

Use the BorderWidth and BorderHeight properties to specify the control's border. The Picture property doesn't affect the skin used unless it contains transparent areas. Use the ForeColor property to change the control's foreground color. Use the Image property to assign a new image to the button (this is displayed on the button's skin not on the control's background). Use the PictureDisplay property to specify how the picture is layered on the control's background. Use the HTMLPicture property to display custom size pictures anywhere in the button's caption, using the <img> tag.

As BackColor property, the Picture property doesn't affect the opaque areas in the current skin. The following sample changes the control's picture on the background :

```
With Button1

   .BorderHeight = 20
   .BorderWidth = 20

   .Picture = LoadPicture("C:\WINNT\Web\tips.gif")
   .PictureDisplay = Tile

   Dim s As String
   s =
"gBHJJGHA5MJAAEIe4AAAFh0OCERiQbigwEobAsXCAljkcHYwDYQkAli0iGAwHYlCA7HZQIp

   s = s +
"mm0IpYjEUogW2JJtCwVBpFIJQKhWYQdAKUwSAwCYGAqSw+DmBRrFYS4YH+AhNiSGYM

   s = s +
"SWBzh4CQMEIQyxSiYAMOILwfBMDSDEAt/DXB0iIFYJAQoXwlBIFqBIZYqhWDGHkAgM4tRa

   s = s +
"YdoNwVAtECHQZ4kxUDrGeAIHwTRaiMB0AsJIxQVguDeCoWQ9B3i/BYJALA1B/gGBwIgeAD

   s = s +
```

```
"cBAAQNQ2BBAeIcFAjgLC+EoAAO4mRzA+DGIsVIAgMBwBABNAJAwAAB1BGBBAOgkAIgC



    .Caption = "&Help;"
    .Image = s
    .VFit = 6
End With
```

Please notice that the only the control's background is affected. Use the Skin property to change the button's skin.

property Button.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 how the Picture property is displayed on the control's background.

Use the Picture property to assign a picture on the control's background. Use the PictureDisplay property to arrange the Picture on the control's background. Use the Image property to assign an image/picture to the button. Use the Style, Skin property to specify the button's visual appearance.

method Button.Refresh ()

Refreshes the control.

Type	Description

Use the Refresh method to refresh the control, if required. Use the BeginUpdate and EndUpdate methods to maintain performance while multiple changes need to be applied to the control.

property Button.RenderType as Long

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

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

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

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

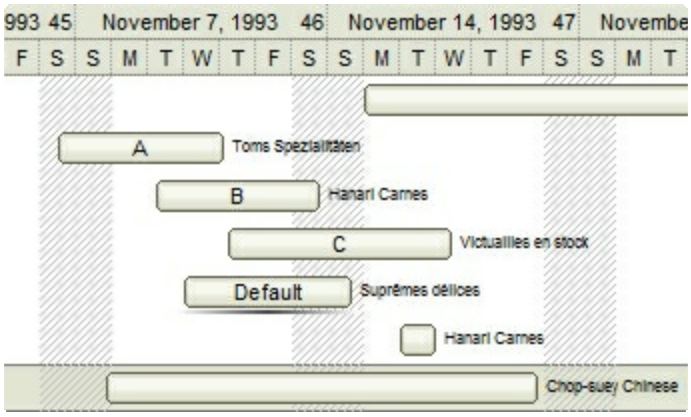
Click here to watch a movie on how you can change the colors to be applied on EBN objects.

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

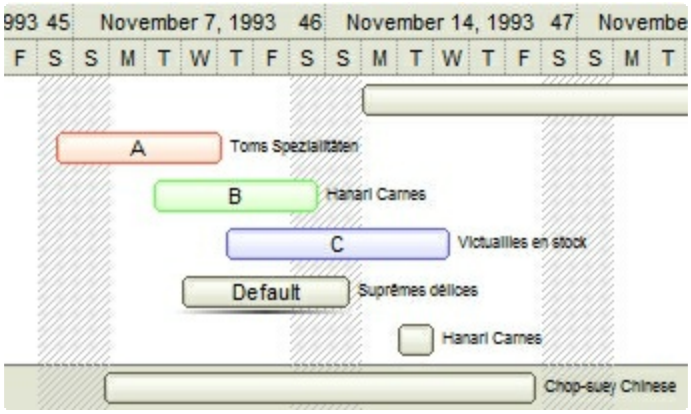
- "A" in Red (RGB(255,0,0), for instance the bar's property exBarColor is 0x10000FF
- "B" in Green (RGB(0,255,0), for instance the bar's property exBarColor is 0x100FF00
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

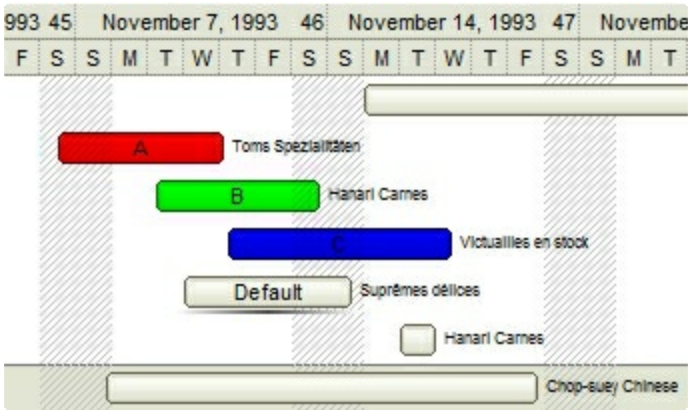
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



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

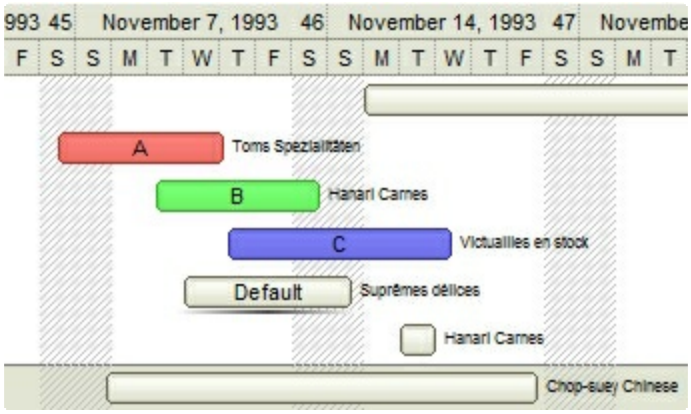


- ***-1**, AND-color scheme*, The color to be applied on the part of the control is an AND bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the AND bit for the entire Blue channel, or in other words, it applies a more Blue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



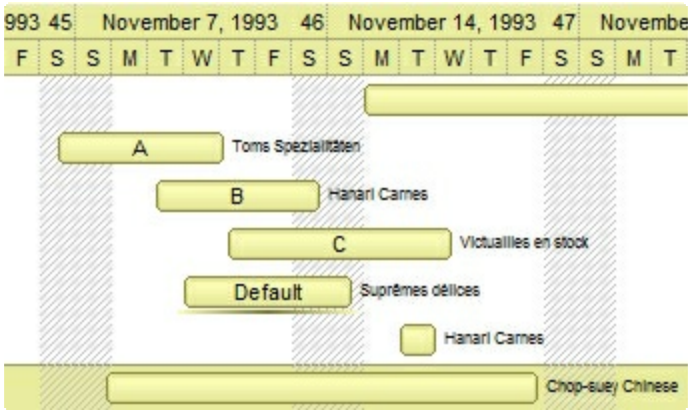
- ***0**, default*, the specified color is applied to the EBN. For instance, the

BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

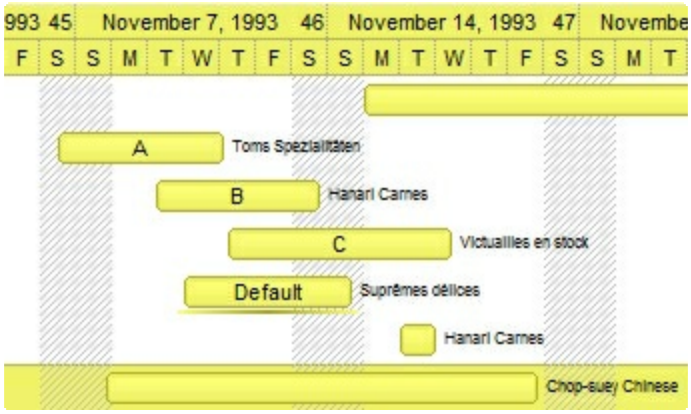


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

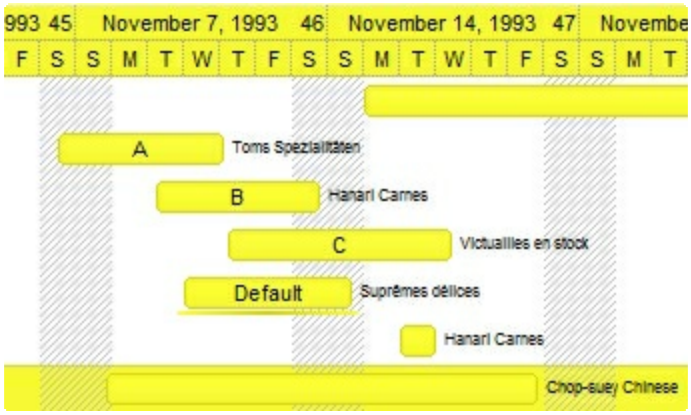
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



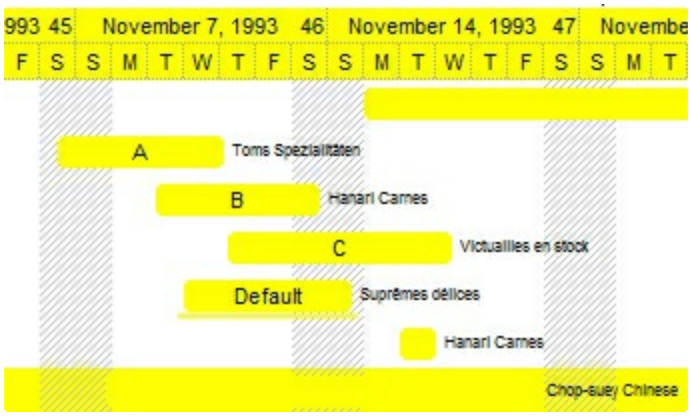
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



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



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



method Button.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 an image list to the control.

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

i = Button1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), in this case the i specifies the index where the icon was added

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

i = Button1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), in this case the i is zero, because the first icon was replaced.

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

Button1.ReplaceIcon 0, i, in this case the i must be the index of the icon that follows to be removed

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

Button1.ReplaceIcon 0, -1

property Button.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 Button.ShowFocusRect as Boolean

Sets or returns a value that determines whether or not the focus rectangle should be shown.

Type	Description
Boolean	A boolean expression that indicates whether the control displays the focus rectangle around the button's caption or image.

By default, the ShowFocusRect property is False. Use the UseFocusSkin and FocusSkin properties to assign a new visual appearance to your button when it has the focus. The following screen shot shows how the thin focused rectangle looks like:

method Button.Skin (State as StateEnum, File as String)

Specifies the skin file to display the specified state.

Type	Description
State as StateEnum	A StateEnum expression that indicates the state skin to be changed.
File as String	A string expression that specifies the path to a skin file (*.ebn), a string expression that indicates the BASE64 encoded string that holds a skin file (*.ebn). Use the Exontrol's exImages tool to build BASE 64 encoded strings on the skin file (*.ebn) you have created. If the File parameter is an empty string the skin is erased, so your button displays only the Image and the Caption of the button without a visual appearance (skin).

The button provides multiple states like: normal, pushed, hot, disabled, custom, focused and so on. Each state has an associated skin that's displayed when certain state occurs. The UseTransparency property specifies whether the control supports transparency. The transparent regions in the control's skin indicates the transparency of the button. Use the Exontrol's exButton Builder to create new skins for your button. Use the ForeColorState property to assign a different foreground color for certain state. Use the UseFocusSkin and FocusSkin properties to change the button's visual appearance when the control has the focus. Use the SkinV method to load EBNs from resources.

Please be aware that Style property changes the following properties, based on the style chosen:

- AllowHotState property
- UseFocusSkin property
- Skin
- FocusSkin

The following sample displays only the exPressed skin when the user clicks the button:

```
With Button1
   .Caption = "&Help"
   .Style = exXPBlue
   .AllowHotState = False
   .UseFocusSkin = False
   .Skin exNormal, ""
End With
```

We would recommend taking a look over the following articles:

- How to build my own skin file?
- How do I assign a skin file to my button?

method Button.SkinV (State as StateEnum, Skin as Variant)

Specifies the skin file to display the specified state.

Type	Description
State as StateEnum	A StateEnum expression that indicates the state skin to be changed.
Skin as Variant	A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file or a string expression that specifies the path to a skin file (*.ebn), a string expression that indicates the BASE64 encoded string that holds a skin file (*.ebn). Use the Exontrol's exImages tool to build BASE 64 encoded strings on the skin file (*.ebn) you have created. If the File parameter is an empty string the skin is erased, so your button displays only the Image and the Caption of the button without a visual appearance (skin).

The SkinV is similar with Skin method, excepts that it can loads the EBN files from safe arrays or in other words from resources. The button provides multiple states like: normal, pushed, hot, disabled, custom, focused and so on. Each state has an associated skin that's displayed when certain state occurs. The UseTransparency property specifies whether the control supports transparency. The transparent regions in the control's skin indicates the transparency of the button.

There are several options to provide EBN files in your project as follows:

- (**path**) The path to the EBN file. This option is useful when your application installs files on the client's machine so you can provide the path to EBN files.

```
With Button1
   .SkinV exNormal, "C:\Program Files\Exontrol\EBN\vistasel.ebn"
End With
```

- (**string**) The BASE64 encoded string that holds the EBN file. This option is useful if you provide EBN objects in the control's Template page or in code. The Exontrol's exImages tool generates BASE64 encoded strings from EBN files.

```
With Button1
   Dim s As String
   s =
"gBFLBCJwBAEHhEJAEGg4BHQDg6AADACAxRDAMgBQKAAzQFAYahyGCGAA
```

```
    s = s +
"SiOKRKEaFYkmiWYwmuIRliOLhBDcKZ6gSl4qDqCokimahqiaJYqk2SYwmyJwg

    .SkinV exNormal, s
End With
```

In order to generate the BASE64 encoded string from your EBN file do the following:

- Run the eXImages tool
- Run the Windows Explorer and select or locate the EBN file. Press the CTRL + C or drop the EBN file in the middle panel (Drag here files such of .bmp, .gif, .ebn, ...)
- The clipboard contains the generated BASE64 string, or you can copy it from the right panel of the eXImages tool. Generally, the string is long, so you can use the s definition to insert it to your code.

- (**array**) A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. This option is useful if you want to provide the EBN files in the project resources. The idea is that you have to provide a safe array of bytes to the Skin parameter of the SkinV method. For instance, the VB6 provides the LoadResData function, the VB/NET or C# provides an internal class Resources where all items in the resources can be accessed through public properties.

  VB6

  ```
  With Button1
    .SkinV exNormal, LoadResData(101, "CUSTOM")
  End With
  ```

  In order to insert the EBN file to the project resources do the following:

  - Click the VB Resource Editor button in the toolbox.
  - Once the VB Resource Editor tool is opened, click the Add Custom Resource ... button in the toolbox
  - Locate, Select the EBN file in the opened file/folder dialog, and press Open button

- The "CUSTOM"\101 item should be inserted in the resource file.
- Click the Save button, so the RES file is being associated with your project.

VB/NET

```
With Exbutton1
   .SkinV(exNormal, WindowsApplication1.My.Resources.vistasel)
End With
```

In order to insert the EBN file to the project resources do the following:

- Select the Project\Properties... from the VS menu
- Click the Resources page
- Click the Add Resource and then Add Existing File...
- Locate, Select the EBN file in the opened file/folder dialog, and press Open button
- The vistasel item is being generated and so it can be accessed in code using: WindowsApplication1.My.Resources.vistasel

C#

```
exbutton1.SkinV(exNormal,
WindowsApplication1.Properties.Resources.vistasel);
```

In order to insert the EBN file to the project resources do the following:

- Select the Project\Properties... from the VS menu
- Click the Resources page
- Click the Add Resource and then Add Existing File...
- Locate, Select the EBN file in the opened file/folder dialog, and press Open button
- The vistasel item is being generated and so it can be accessed in code using: WindowsApplication1.Properties.Resources.vistasel

property Button.State as StateEnum

Specifies the control's state.

Type	Description
StateEnum	A StateEnum expression that indicates the button's state.

Use the State property to specify the button's state. Use the Mode property to change the button's way to handle the State property. For instance, if the Mode property is exCustom you can control what state being displayed when user clicks the button. The control fires Click event when user clicks a button. The control fires the StateChange event when the State property is changed. Use the Enabled property to disable the button. Use the Caption property to assign a caption to your button. Use the Image property to assign an image to your button.

property Button.Style as StyleEnum

Sets or returns a value that determines how the button will be drawn.

Type	Description
StyleEnum	A StyleEnum expression that indicates the button's predefined state.

By default, the Style property is exDefault. Use the Style property to change the button's visual appearance to a predefined value. Use the Skin, FocusSkin methods to change the button's visual appearance for a certain state. Use the UseTransparency property to specify round corners for the button, as defined by EBN object being displayed. Use the BackColor property to specify the control's background color.

Please be aware that Style property changes the following properties, based on the style chosen:

- AllowHotState property
- UseFocusSkin property
- Skin
- FocusSkin

We would recommend checking the following articles:

- How to build my own skin file?
- How do I assign a skin file to my button?

Also, you can view the following items:

- Create a complex EBN file in 1 minute.
- Using the Exontrol's ExPropertiesList to browse EBN objects.

property Button.Template as String

Specifies the control's template.

Type	Description
String	A String expression that specifies the initialization code (x-script language).

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

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

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

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

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

An instruction can be one of the following:

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

of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))

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

The Template supports the following general functions:

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

The following sample initializes the control's image and caption at design mode, using the Template page:

```
' Specifies the control's caption
Caption = "Exontrol's exButton component is our answer to your GUI needs"
WordWrap = True

' https://www.exontrol.com/sg.jsp?content=support/faq#eximages
Image =
"gBHJJGHA5MIwAEIe4AAAFhwbiAliQwig7ixFjBQjRbjhljxwkB7kSFkiQkyblCllSwli7lzFmDQm1
```

The button's visual appearance in this case will be:

property Button.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 Button.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:

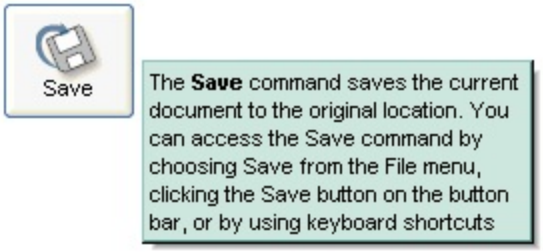
- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **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 Button.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. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.



property Button.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

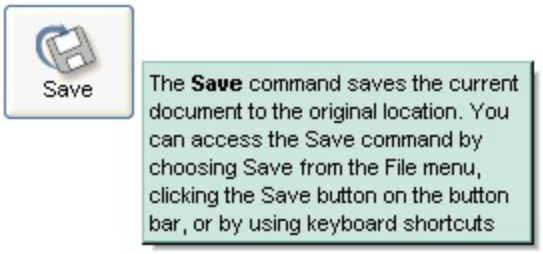
Use the ToolTipFont property to assign a font for the control's tooltip. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. You can use the <*font*> HTML element, in the tooltip's description to assign a different font for portions of text. Use the ToolTipText property to assign a tooltip to your button. Use the <a id;tooltip> element to assign a custom tooltip for a specified anchor element.

property Button.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. Use the ToolTipDelay property specifies the time in ms that passes before the ToolTip appears.



property Button.ToolTipText as String

Specifies the control's tooltip text.

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

Use the ToolTipText and ToolTipTitle properties to define the button'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 inside the button's tooltip.

The ToolTipText supports the following HTML tags:

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

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

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

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

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

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a

known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

In the 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 Button1
    .Object.ToolTipText = "In VB, this is the right way to call the ToolTipText property at runtime."
End With
```

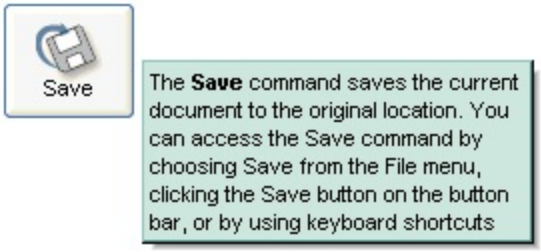
If the Object property is missing, the code changes the TooltipText property of the VB extended class, instead calling object's property.

property Button.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 button's tooltip. Use the ToolTipDelay and ToolTipPopDelay properties to specify the time in ms that passes before the ToolTip appears.



property Button.ToolTipWidth as Long

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

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

Use the ToolTipWidth property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipFont property to assign a font for the control's tooltip.

property Button.UseFocusSkin as Boolean

Specifies whether the focus skins are used when control has the focus.

Type	Description
Boolean	A boolean expression that indicates whether the button provides a different visual appearance when the button has the focus.

Use the UseFocusSkin property to change the visual appearance for your button when it has the focus. Use the FocusSkin method to assign a skin for the button when it has the focus. Use the Style property to assign a predefined appearance to your button. Use the ShowFocusRect property to draw a thin rectangle around the button's caption when it has the focus.

Please be aware that Style property changes the following properties, based on the style chosen:

- AllowHotState property
- UseFocusSkin property
- Skin
- FocusSkin

If you require certain value for the UseFocusSkin property you have to change the UseFocusSkin property after changing the Style property.

For instance, the exMAC style provides a skin when the button has the focus.

property Button.UserData as Variant

Gets or sets the user-definable data for the current object.

Type	Description
Variant	A Variant value that specifies the control's user data.

Use the UserData property to associate an extra data to your button. The UserData is not used in any way by the control. Use the Caption property to assign a caption to the button. Use the Image property to assign an icon, picture to your button.

property Button.UseTransparency as Boolean

Specifies whether the control supports transparency.

Type	Description
Boolean	A boolean expression that indicates whether the control supports transparency.

By default, the UseTransparency property is True. The UseTransparency property specifies whether the control supports transparency. The transparent regions in the control's skin indicates the transparency of the button. If the button's skin contains no transparent regions, the UseTransparency property has no effect. Use the Skin method to assign a skin to the button's face. Use the BackColor property to change the control's background color.

The following screen shot displays the button, when the UseTransparency property is True:

The following screen shot displays the button, when the UseTransparency property is False (check the white corners of the button):

The following template assigns the skin  to the normal state, and the skin  to the pushed state.

```
BeginUpdate()
Style = -1

Skin(0,
"gBFLBCJwBAEHhEJAEGg4BOoJg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUAIIQLGE



Skin(1,
"gBFLBCJwBAEHhEJAEGg4BbQKg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUAIIQLG



IncClientState(1,0) = 0
IncClientState(1,1) = 0
```

```
UseTransparency = True
EndUpdate()
```

property Button.VAlignment as VAlignmentEnum

Specifies the caption's vertical alignment.

Type	Description
VAlignmentEnum	A VAlignmentEnum expression that indicates the vertical alignment of the button's caption.

Use the Caption property to assign a caption to the button. By default, the VAlignment property is exMiddle. Use the VAlignment property to change the vertical alignment for the caption. Use the ImageVAlignment property to change the vertical alignment for the image. Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area

property Button.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 Button.VFit as Long

Specifies a value that indicates the vertical offset to fit image with the caption.

Type	Description
Long	A long expression that indicates the vertical offset to fit image with the caption.

By default, the HFit property is 0(zero). Use the VFit, HFit, IncClientSide properties to adjust the control's client area and to organize the image and the caption positions in the control's client area.

property Button.WordWrap as Boolean

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

Type	Description
Boolean	A boolean expression that indicates whether a multiline text automatically wraps words to the beginning of the next line when necessary.

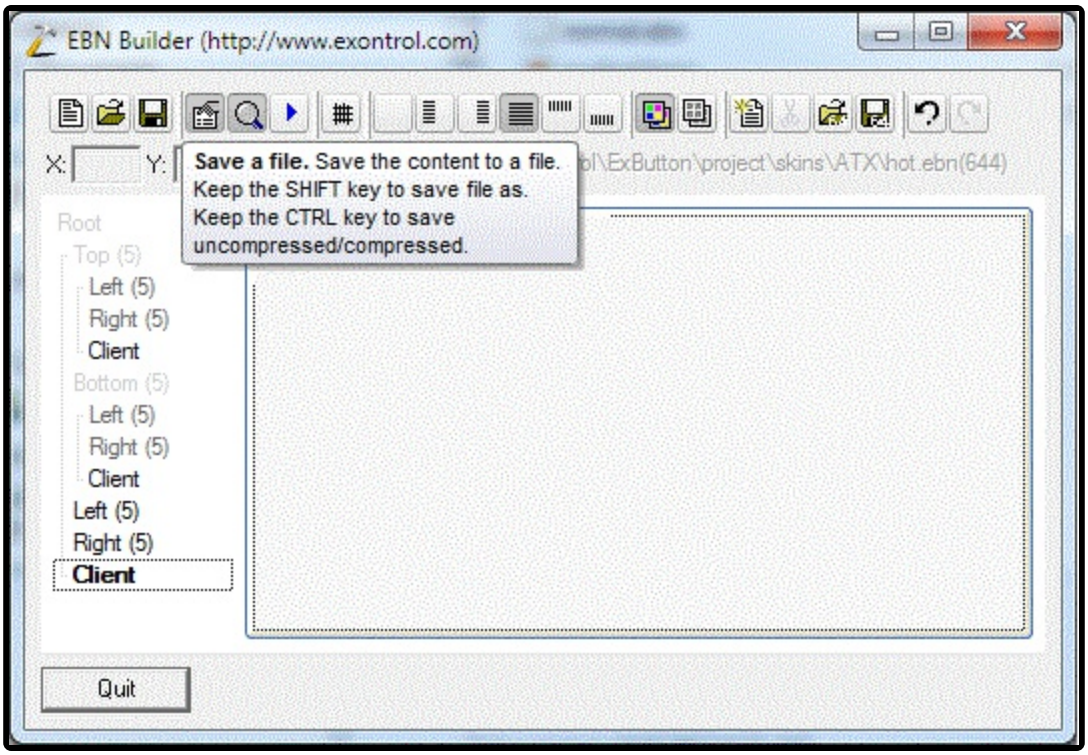
Use the Caption property to assign a caption to your button.

ButtonBuilder 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: {91C7CB9E-7B3C-406E-9399-D8F344CABD7B}. The object's program identifier is: "Exontrol.ButtonBuilder". The /COM object module is: "ExButton.dll"

Exontrol's exButton lets you the ability to create and use your own look and eel buttons. The ButtonBuilder tool helps you to create or change skin file to be displayed in the exButton control. The ButtonBuilder tool acts like an ActiveX control that can be placed on any container that supports ActiveX containment. The ButtonBuilder tool provides a button area, toolbars, a magnify window and a picture properties window. The ButtonBuilder component contains everything you need to build and create new skin files for your button. The ButtonBuilder component saves everything that a skin required like, pictures, properties and attributes for each area defined in the skin. It compresses the information so the size of the skin file will be as less as possible (also, it supports saving the skin files in uncompressed format). Use the Exontrol's exImages tool to convert you skin files to BASE64 encoded string. Even if it is not a requirement the extension of the skin files for Exontrol's exButton component is **ebn**.

Use the Skin, FocusSkin methods to assign a skin file or a BASE64 encoded string that holds a skin file to your button to change the button's visual appearance for a specified state.



We would recommend taking a look over the following articles:

- How to build my own skin file?
- How do I assign a skin file to my button?

The ButtonBuilder object supports the following properties and methods:

Name	Description
Load	Loads the button skin from a file.

method ButtonBuilder.Load (FileName as String)

Loads the button skin from a file.

Type	Description
FileName as String	A String expression that indicates the path to a skin file (*.ebn)

The skin file must be created using the Exontrol's exButton Builder. The same file can be passed to Skin or FocusSkin methods as well.

Button 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: {91C7CB9E-7B3C-406E-9399-D8F344CABD7B}. The object's program identifier is: "Exontrol.ButtonBuilder". The /COM object module is: "ExButton.dll"

The exButton control is designed to enhance your Windows-based programs by offering the look-and-feel of past and present GUI design elements. The Exontrol's Button object supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user presses and then releases the left mouse button over the control.
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.
StateChange	Occurs when the button's state is changing.

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 button 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;tooltiptext>anchor*, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "tooltiptext". 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,
AxEXBUTTONLib._IButtonEvents_AnchorClickEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure AnchorClick(sender: System.Object; e:
AxEXBUTTONLib._IButtonEvents_AnchorClickEvent);
begin
end;
```

Powe…

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

VB.NET

```
Private Sub AnchorClick(ByVal sender As System.Object, ByVal e As
AxEXBUTTONLib._IButtonEvents_AnchorClickEvent) Handles AnchorClick
End Sub
```

VB6

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VBA

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VFP

```
LPARAMETERS AnchorID,Options
```

Xbas…

```
PROCEDURE OnAnchorClick(oButton,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

Use the Click event to notify your application that user clicks the button. Also, the Click event is fired if the user releases the SPACE key while the button has the focus. Use the KeyDown event to avoid changing the button's State while user presses the SPACE key (handle the KeyDown event and pass 0 to KeyCode parameter). If your Windows scheme swifts the left button with the right button the Click event is fired when user releases the right mouse button over the control. Use the Click event to change the button's State when the button's Mode property is exCustom.

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

C#

```
private void Click(object sender)
{
}
```

VB

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

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

C#

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

C++

```
void OnClick()
{
}
```

C++ Builder

```
void __fastcall Click(TObject *Sender)
{
}
```

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event Click()
end event Click
```

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnClick(oButton)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComClick
   Forward Send OnComClick
```

```
End_Procedure
```

Visual Objects

```
METHOD OCX_Click() CLASS MainDialog
RETURN NIL
```

X++

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

XBasic

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

dBASE

```
function nativeObject_Click()
return
```

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

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

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

The DblClick event is fired when the user dbl clicks on the control. Use the DblClick event to notify your application that user double clicked the button. Use the Click event to notify your application that the user clicks the button.

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,
AxEXBUTTONLib._IButtonEvents_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:
AxEXBUTTONLib._IButtonEvents_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
AxEXBUTTONLib._IButtonEvents_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(oButton,Shift,X,Y)
RETURN
```

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

Java…

```
<SCRIPT EVENT="DblClick(Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DblClick(Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDblClick Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
   Forward Send OnComDblClick llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_DblClick(Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DblClick(int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function DblClick as v (Shift as N,X as OLE::Exontrol.Button.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Button.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:

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

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

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

C#
```
private void KeyDownEvent(object sender,
AxEXBUTTONLib._IButtonEvents_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:
AxEXBUTTONLib._IButtonEvents_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
AxEXBUTTONLib._IButtonEvents_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(oButton,KeyCode,Shift)
RETURN
```

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

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

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

Visual Data…
```
Procedure OnComKeyDown Short llKeyCode Short llShift
   Forward Send OnComKeyDown llKeyCode llShift
End_Procedure
```

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

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

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

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

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

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

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

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,
AxEXBUTTONLib._IButtonEvents_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:
AxEXBUTTONLib._IButtonEvents_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
AxEXBUTTONLib._IButtonEvents_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(oButton,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,
AxEXBUTTONLib._IButtonEvents_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:
AxEXBUTTONLib._IButtonEvents_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
AxEXBUTTONLib._IButtonEvents_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(oButton,KeyCode,Shift)
RETURN
```

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

Java…

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

VBSc…

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

```
</SCRIPT>
```

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

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

Occurs when the user presses a mouse button.

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

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

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

C#

```
private void MouseDownEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseDownEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseDownEvent
End Sub
```

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

C#

```
private void MouseDownEvent(object sender,
AxEXBUTTONLib._IButtonEvents_MouseDownEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure MouseDownEvent(sender: System.Object; e:
AxEXBUTTONLib._IButtonEvents_MouseDownEvent);
begin
end;
```

Powe…

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

VB.NET

```
Private Sub MouseDownEvent(ByVal sender As System.Object, ByVal e As
AxEXBUTTONLib._IButtonEvents_MouseDownEvent) Handles MouseDownEvent
End Sub
```

VB6

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

VBA

```
Private Sub MouseDown(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseDown(oButton,Button,Shift,X,Y)
RETURN
```

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

Java…

```
<SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseDown(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseDown Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseDown llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseDown(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseDown(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseDown as v (Button as N,Shift as N,X as
OLE::Exontrol.Button.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Button.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseDown(Button,Shift,X,Y)
return
```

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

Occurs when the user moves the mouse.

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

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its margins.

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

C#

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

VB

```
Private Sub MouseMoveEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseMoveEvent
End Sub
```

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

C#

```
private void MouseMoveEvent(object sender,
AxEXBUTTONLib._IButtonEvents_MouseMoveEvent e)
{
}
```

C++

```
void OnMouseMove(short Button,short Shift,long X,long Y)
```

```
{
}
```

C++ Builder

```
void __fastcall MouseMove(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

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

Delphi 8 (.NET only)

```
procedure MouseMoveEvent(sender: System.Object; e:
AxEXBUTTONLib._IButtonEvents_MouseMoveEvent);
begin
end;
```

Powe…

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

VB.NET

```
Private Sub MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXBUTTONLib._IButtonEvents_MouseMoveEvent) Handles MouseMoveEvent
End Sub
```

VB6

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

VBA

```
Private Sub MouseMove(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseMove(oButton,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.Button.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Button.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

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

Occurs when the user releases a mouse button.

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

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

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

C#

```
private void MouseUpEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseUpEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseUpEvent
End Sub
```

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

C#

```
private void MouseUpEvent(object sender,
AxEXBUTTONLib._IButtonEvents_MouseUpEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure MouseUpEvent(sender: System.Object; e:
AxEXBUTTONLib._IButtonEvents_MouseUpEvent);
begin
end;
```

Powe…

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

VB.NET

```
Private Sub MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXBUTTONLib._IButtonEvents_MouseUpEvent) Handles MouseUpEvent
End Sub
```

VB6

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

VBA

```
Private Sub MouseUp(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseUp(oButton,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.Button.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Button.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event StateChange ()

Occurs when the button's state is changing.

Type	Description

Use the StateChange event to notify your application that the button changes its state. The State property determines the button's state. Use the AllowHotState property to allow hot state for your button. Use the Skin method to assign a skin to your button.

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

C#

```
private void StateChange(object sender)
{
}
```

VB

```
Private Sub StateChange(ByVal sender As System.Object) Handles StateChange
End Sub
```

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

C#

```
private void StateChange(object sender, EventArgs e)
{
}
```

C++

```
void OnStateChange()
{
}
```

C++ Builder

```
void __fastcall StateChange(TObject *Sender)
{
}
```

Delphi

```
procedure StateChange(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

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

Powe…

```
begin event StateChange()
end event StateChange
```

VB.NET

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

VB6

```
Private Sub StateChange()
End Sub
```

VBA

```
Private Sub StateChange()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnStateChange(oButton)
RETURN
```

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

Java…

```
<SCRIPT EVENT="StateChange()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

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

Visual Data…

```
Procedure OnComStateChange
   Forward Send OnComStateChange
End_Procedure
```

Visual Objects

```
METHOD OCX_StateChange() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_StateChange()
{
```

```
}
```

XBasic

```
function StateChange as v ()
end function
```

dBASE

```
function nativeObject_StateChange()
return
```

How to replace my old VB buttons with this new button?

Actually, replacing old VB Command buttons with the Exontrol's exButton component is just like replacing a string with another.

You have the following options to replace the VB Command objects with the Exontrol's exButton component.

A)

- Make the first step to add the ExButton to the VB project like any control with the project open (Ctrl+T), save and close it.
- Do the "VB.Command" search and replace in each frm file.

B)

- Open the Project1.vbp and add a reference to the exbutton.dll by adding the line **~Object={65D9132C-B295-42A0-8421-B8B1DA27C5CE}#1.0#0; ExButton.dll~** Do not include the ~ characters.
- Open the Form1.frm and add the **~Object = "{65D9132C-B295-42A0-8421-B8B1DA27C5CE}#1.0#0"; "ExButton.dll"~** reference to the form just before 'Begin VB.Form'
- Replace all **~VB.CommandButton~** with the **~EXBUTTONLibCtl.Button~** and save the form1.frm file.

Now, the idea to replace the old VB button with the new button is to use a tool that's able to find and replace strings. For instance if you have multiple vbp files adding the object reference to the files could be like follows. Look for a property in the .vbp file that could be located in all your vbp files, like VersionCompanyName (or whatever) and do Replace('VersionCompanyName="YourCompanyName"', 'VersionCompanyName="YourCompanyName"\r\nObject={65D9132C-B295-42A0-8421-B8B1DA27C5CE}#1.0#0; ExButton.dll'. This way you actually added the reference to the project file. The same for the file reference in the frm files where the 'Begin VB.Form' could be located.

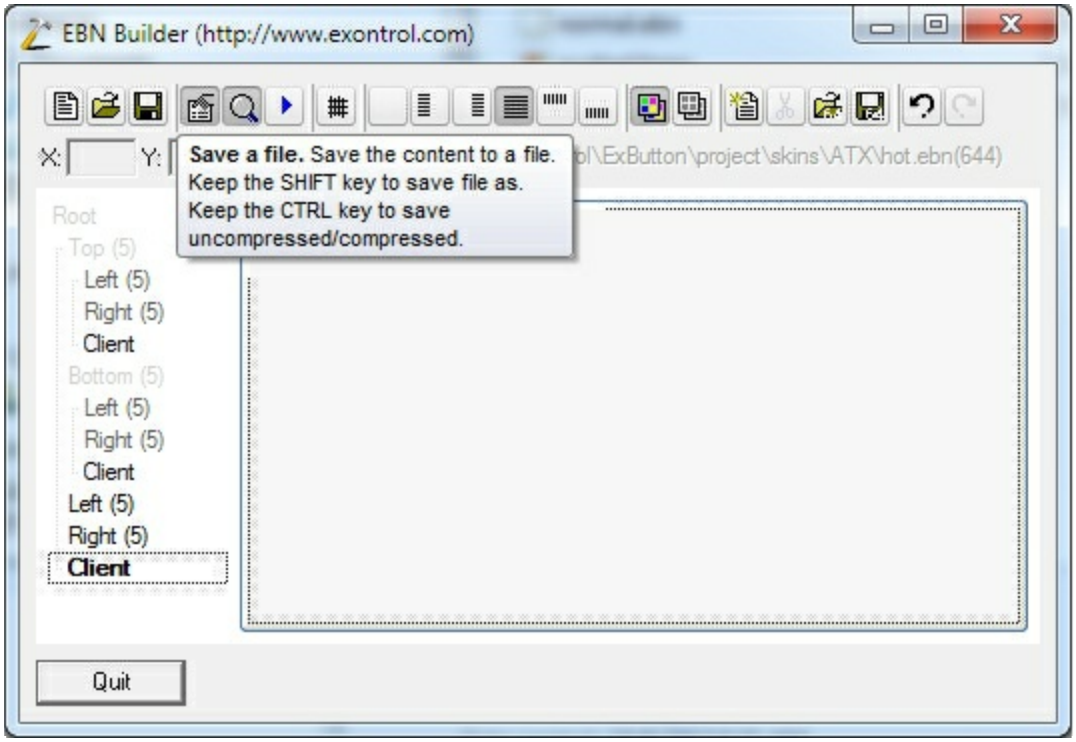
How to build my own skin file?

The Exontrol ExButton library installs the ButtonBuilder component ("Exontrol.ButtonBuilder" is the control's identifier) that helps you to build new skin files for your buttons. Before showing how you can build your own skin file, we have to review for a bit how the ButtonBuilder can be used. If you already know how to use the ButtonBuilder component click here to see how to start your own skin file (visual appearance for a specified state of the button).

- If you are a VB developer, click the 'Toolbox' panel, and choose the 'Components' from the panel's context menu. Once that you check and apply the 'ExButton 1.0 Control Library' from the opened dialog, your 'Toolbox' panel includes two new components: Button and ButtonBuilder.
- If you are a VC developer, select the dialog where you want to insert the component, select 'Insert ActiveX Control' from its context menu, and dbl click the 'ExButton ActiveX Control' item in the opened dialog.

Once that you have inserted a ButtonBuilder component to a form or dialog, you are ready to build new skin files for your buttons. Of course, you can use the ButtonBuilder component to load and change already saved skin files. When the form that contains a ButtonBuilder component is opened, the ButtonBuilder component automatically shows the 'Zoom' and 'Properties' panels. The 'Zoom' panel helps user to magnify different portions of the screen. The 'Properties' panel contains information like, background color, background picture for the selected object.

The skin method, in it's simplest form, uses a single graphic file assigned to the client area of the button. 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 button. The skin file is organized as a hierarchical list of objects. Each object can display a portion of picture with attributes like tile, stretch or transparent or a background color. The position for each child object is relative to its parent, and can be aligned to any side of the parent's client area. Important to notice is that the position of the objects can use simple arithmetic operation like 10 + 50% that means 10 pixels plus a half of parent's size, or 100% that means 100% of the parent size.



The ButtonBuilder's toolbox contains the following buttons:

Action	Description
New file	Creates an empty skin file. The builder automatically shares the skin area in multiple pieces that identifies the client area of the skin and the corners.
Open a file	Opens a file. The file should be saved previously using the Save button. By default, the builder loads *.ebn files.
Save a file	Saves the skin to a file. The builder saves everything that's required for the skin. *Save As , if you are pressing the SHIFT key while clicking the 'Save' button you can choose a new file where to save the skin. By default, the builder compresses the files, so you can press CTRL key to save files uncompressed (the message ' Save (uncompressed) as 'should appear on the save file dialog).*
Properties	A check button that indicates whether the 'Properties' panel is visible or hidden. The 'Properties' panel holds information about background of the selected object. Also, the the 'Properties' panel contains the list of pictures used by the skin. Details here.
	A check button that specifies whether the 'Zoom' panel is visible or hidden. You can use the 'Zoom' panel to magnify different portions of screen. In order to visualize a specified

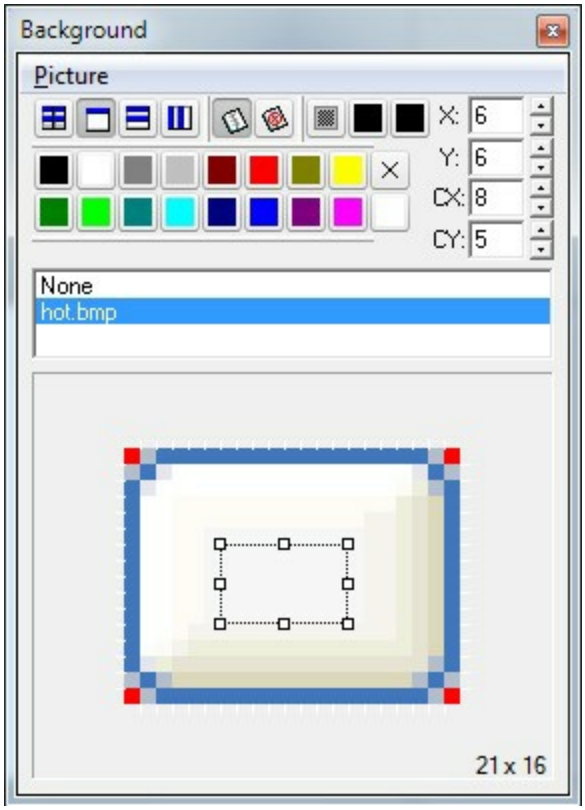
Zoom	portion of the screen you can press "CTRL" key while moving the mouse, or you can click into the 'Zoom' window and drag the focused rectangle to the area being magnified. You can magnify the are by keeping the (SHIFT +)CTRL key and clicking the Zoom Window.
Test	Shows the current EBN object to a separate window. Opens a new window where you can see running the exButton using the State exNormal as defined by the skin. Keep the CTRL key to assign a new skin for the exPushed state.
Draw grid lines	A check button that indicates whether the ButtonBuilder draws the grid lines around the objects in the skin. The grid lines are not painted in the Test window.
Alignment	A set of six radio buttons that indicates the object's alignment relative to its parent. The list of radio buttons in their listed order is: None, Left, Right, Client, Top and Bottom. For instance, if an object has the Left Alignment, it means that the object shares the left area of the parent with itself. The object's coordinates are defined in the edit controls labeled: X, Y, CX and CY. The edit controls that handle coordinates are enabled based on the object's alignment. For instance, if the object's alignment is left, only the CX coordinate will be enabled, or if the object's alignment is None, then all coordinates are enabled. The coordinates are relative to the parent object, and they may contain arithmetic expressions, and % sign (percent indicates that the object is % from the size of its parent) as well.
Colorable Object	The object is colorable, which means that the EBN color is applied to this part when the color is applied to the entire EBN object. For instance, at runtime the identifier 0x1FF0000 applies blue color to all parts that compose the EBN object, including the selected object.
Not-Colorable Object	The object is not-colorable, which means that the EBN color is not applied to this part when the color is applied to the entire EBN object. For instance, at runtime the identifier 0x1FF0000 applies blue color to all parts that compose the EBN object, excluding the selected object.
	Creates a new child object. The newly created object is child of the selected object. By default, the newly created object has no picture or color associated to it. You have to

Insert Object	define the object's background using the 'Properties' panel. You can also, insert a new child object while ButtonBuilder is focused by pressing the 'Insert' key.
Remove Object	Removes the selected object. Also, you can remove the selected object by pressing the 'Delete' key. Note, that the client area of the skin (the area that displays the 'Caption' string) is not allowed to be removed.
Load BASE64	Loads from the clipboard a BASE64 string that encodes an EBN object.
Save BASE64	Generates the BASE64 string that encodes the current EBN object, and copies it in the clipboard as text.
Undo	Restores the last operation.
Redo	Reverse of the Undo operation.

Notes:

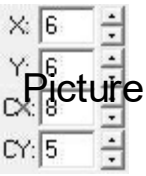
- the selected object is always marked using markers.
- any change in the 'Properties' panel will be reflected in the selected object.
- use 'Up' key to select the parent of the selected object, while the ButtonBuilder is focused.
- use the 'Down' key to select the first child of the selected object, while the ButtonBuilder is focused.
- use the 'Left' or 'Right' key to move through the objects that have the same parent as selected object.
- you can change the order of the objects (that have the same parent) using combination of CTRL + PgUp or CTRL + PgDn key.

The 'Properties' panel is a resizable window that's visible only if the button is pressed. The caption of its window is 'Background'. The 'Properties' panel is always updated when the selected object is changed. The 'Properties' panel defines the list of pictures used by the skin. The 'Properties' panel looks like follows:



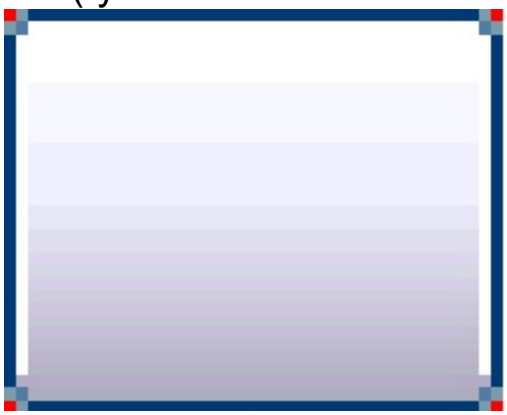
Use the Picture menu to insert, delete a picture object from the skin file. Note that all picture files are saved to the skin file (ebn file), no matter if they are used or not. The ButtonBuilder compresses the file, so even if you are using a bitmap file or a gif file, the file of the skin will be compressed (use the CTRL key to save the file as uncompressed). The 'Properties''s toolbox contains the following buttons (in their order):

Action	Description
Tile, Stretch, Horizontal or Vertical Stretch	A set of two radio buttons that defines the way how the picture is displayed on the selected object: tiled, stretch, horizontal stretch or vertical stretch.
Transparent	A set of three radio buttons that defines the picture's transparency. If the first button is pressed, the picture is opaque, so no transparent colors are used. If the second button is pressed, the picture is transparent. No picture or background is applied to the selected object. If the third button is pressed, the last two buttons (the black buttons) define the transparent color from and transparent color to. In order to select a new transparent color, you have to click on the one of the last two buttons and drag to the desired color. Once that you have selected a transparency color, you have to presses the button again to apply the transparent color to the selected object.
	Defines a set of predefined colors. The **X** button clears the background color of the selected object. The bottom-right

 Colors	button (bellow to **X** button), helps user to add a custom color. **How?** Click the button and drag to desired color. Once that you have selected a custom color, you can press the button again and the builder will apply the selected color to the selected object.
 Picture coordinates	The X, Y, CX, CY edit controls define the coordinates of the picture on the background of the selected object.

Now, that we are ready to go, we can start building the skin for the button. If you have already a skin file check the How to assign a skin file to my button? Between steps you can save the skin file using the Save button.

1. **Choosing the picture files** that we are using to build our button. You can have a BMP file, a GIF file or a JPEG file (or any picture file that your Windows recognize), though we prefer the BMP file since it holds information about the picture without losing colors by compressing. In our case we choose this one  (you can save it as a BMP file).

   Here's the same picture but this time it is zoomed: 
2. **Loading the picture file** using the **Background** panel, by choosing the Picture\ Add New item menu. Please notice that the selected object in the skin is the skin's client area (the area where the 'Caption' string is painted, so the picture will be applied as tile on the skin's client area. In the same time, we can specify the skin's client area by changing the coordinates of the picture even by clicking the slider buttons on the coordinates panel, or by dragging the selected area in the picture. We have two options to avoid displaying that picture on the skin's client area. One by clicking the **None** item in the pictures list, and this way we ensure that the skin's client area ha no picture associated, or clicking the Transparent button. In this case, the skin's client area have a picture associated but it is not painted. At this point you can download the skin file here.
3. **Defining the corners and margins for the button.** By default, the builder creates few objects that specify the corners and margins of the button. The size for these objects can be changed any time. By default, the size of the corners is 4 (pixels). In our case if we take a closer look using the **Zoom** window, at the picture we have chosen we will see that 4 pixels are quite enough for margins and corners, but we will

change the margins as follows: the left, right and top margins set to 2 pixels, and the bottom margin set to 3 pixels. This way we will ensure that our corners and borders are correct defined. At this point you can download the skin file here.

4. **Defining the look and feel for each object in the skin.** Now that all parts in our skin are well defined, we will take each part that's displayed and we will define how it will look like. For instance, we will define the left-top corner (the one with a red pixel), but the same thing can be done for all objects in the skin. So, first we have to select the object by clicking inside the object until builder marks the selected object. If your object is too small press the CTRL key and the area will be zoomed in the Zoom window (also the cursor will be marked by a cross), or use the arrow keys until you get the desired object. After you selected the object the Zoom window automatically magnifies the object in the Zoom window for a better view. If you feel that grid lines that's painted to mark the objects in the skin, just press the Draw grid lines button ▦. Go to Background panel and click the normal.bmp item. This way you assign a picture to the selected object. Chose the way how the picture is displayed: tile, stretch, transparent and adjusts the picture coordinates until you get the desired image in the selected object. If your object require transparent areas you need to press the Transparency button ▩ and choose the taransparent color "from" and "two" using the two buttons right to the Transparency button (click and drag the cursor over the color you want to select, you can drag the cursor directly to the zoom window where the screen is magnified). After you select the transparency colors the builder considers the colors being the range of transparency used. Click once again one of these two buttons, and the transparency will be applied to the selected object. At this point you can download the skin file here.
5. **After all objects were defined** click the Save button 💾 and your skin is ready to be used. At this point you can download the skin file here. How to assign a skin file to my button?

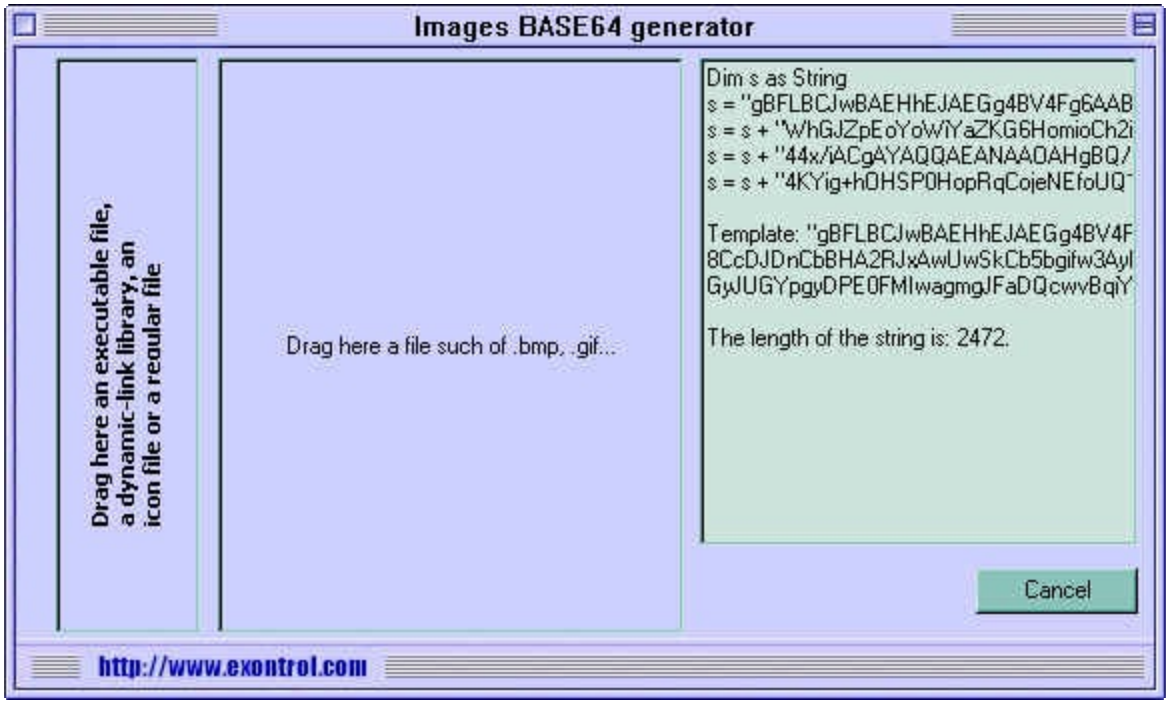
How do I assign a skin file to my button?

We assume that you have already a skin file (else you can search for *.ebn files in the samples folder of the Exontrol's ExButton).

There are two options to load your skin file to the button using the Skin or FocusSkin method of the ExButton:

- converting the skin file to a BASE64 encoded string.
- loading directly the file

Converting the skin file to a BASE64 encoded string is possible using the Exontrol's ExImages tool. It is free to use, so you have to download it and to run the ExImages.exe file. The following screen shot shows you the mainframe from the Exontrol's ExImages tool:



Open a Windows Explorer, locate your skin file and drag the file over the area **"Drag here a file such of .... "**. The tool generates the BASE64 encoded string in two formats: VB or Template.

The VB format shows like follows:

```
Dim s as String
s =
"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIX
QKMUIxVAcLQxCgCYRhYABRiUAoJkjkMYhSDOFgzARHcxRPDgARrDyZQAkOQ5FDGFo+
ShFQxTRC9CQpHaEYqkeA3fgmTYXTxJM7yfQVFxlCwTIwFGQqJgmVpPABYERyWKoSzJMy
ERpGCyIDqqbJXVxFYj3DCscw/KIYaqlGS5Ni+IZ2TLNMz4BAdEQfKSEaAgOToboaE5GB5G
eRRbT1HYtKDEcQhepIbpaH5YQjkMBibBNZ4pAavcroeK7FqeI5ua7ach5fisB5EAARYREGr
cEqPGZ5ShjGJ1MK0CxzIwDboBPbNdwXP56cIAAx8IJbD0GJQGoIQ1jgGAbhmTZXGsLZ7
```

and it can be copied and pasted to your VB, VC code.

The Template form looks like:

```
Template:
"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIX
QKMUIxVAcLQxCgCYRhYABRiUAoJkjkMYhSDOFgzARHcxRPDgARrDyZQAkOQ5FDGFo+
ShFQxTRC9CQpHaEYqkeA3fgmTYXTxJM7yfQVFxlCwTIwFGQqJgmVpPABYERyWKoSzJMy
ERpGCyIDqqbJXVxFYj3DCscw/KIYaqlGS5Ni+IZ2TLNMz4BAdEQfKSEaAgOToboaE5GB5G
eRRbT1HYtKDEcQhepIbpaH5YQjkMBibBNZ4pAavcroeK7FqeI5ua7ach5fisB5EAARYREGr
cEqPGZ5ShjGJ1MK0CxzIwDboBPbNdwXP56cIAAx8IJbD0GJQGoIQ1jgGAbhmTZXGsLZ7
AsTpKDEVoIG0QAaJyA4bleZgCiEJpjHmSJaGENgLgwRpTgUCAhAMEIEICSZ+EUAxkCQKB
```

and it can be used in Template pages. Also, the tool displays the length of the string that's required to hold the file you dragged. Important to notice is that the ExImages tool compresses the file before generating the BASE64 encoded string, but converting it to a BASE64 string it means that the size of the string will be with 1/4 greater than compressed file. The BASE64 encoded strings are useful to hold your icons, pictures, skins in string instead adding all kind of files to your application.

So, after we get the BASE64 encoded string all that you need to do is to pass the s variable to Skin, or FocusSkin method like in the following sample:

```
With Button1

   Dim s As String
   s =
"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIX
QKMUIxVAcLQxCgCYRhYABRiUAoJkjkMYhSDOFgzARHcxRPDgARrDyZQAkOQ5FDGFo+
ShFQxTRC9CQpHaEYqkeA3fgmTYXTxJM7yfQVFxlCwTIwFGQqJgmVpPABYERyWKoSzJMy
ERpGCyIDqqbJXVxFYj3DCscw/KIYaqlGS5Ni+IZ2TLNMz4BAdEQfKSEaAgOToboaE5GB5G
```

The sample shows how to apply a skin to the exNormal state of the button. Using the same way you can assign skin for any state of the button, and you can assign skins for states that you are going to use, not for all. For instance, if you have AllowHotState property is False, the exHot state should not be assigned (because it is never used). Also, if you have UseFocusSkin property on True, instead calling the Skin method you should call FocusSkin method. Also, if multiple skins are applied in the same time we would recommend using the the BeginUpdate and EndUpdate methods. You will find a simple sample bellow this page.

Loading directly the file is possible by passing the path to the skin file to the Skin or FocusSkin method like in the following sample:

```
With Button1
   .Skin exNormal, "D:\Exontrol\ExButton\project\skins\XPSilver\normal.ebn"
End With
```

Use the BeginUpdate and EndUpdate methods to avoid painting the button while adding skins for each state of the button like in the following sample:

```
With Button1
   .BeginUpdate

      Dim strPath As String
      strPath = "D:\Exontrol\ExButton\sample\VB\Builder\Predefined\XPSilver\"

      .UseFocusSkin = True
      .Skin exNormal, strPath + "normal.ebn"
      .Skin exHot, strPath + "hot.ebn"
      .Skin exPushed, strPath + "pushed.ebn"
      .FocusSkin exNormal, strPath + "focus.ebn"
```

```
    .EndUpdate
End With
```