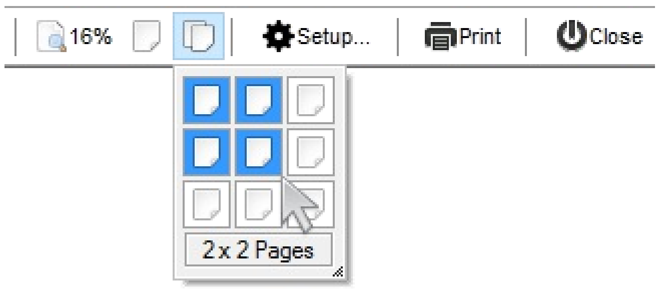
ExToolBar.CRD

The eXToolBar.CRD component is a graphical control element on which on-screen buttons, icons, menus, or other input or output elements are placed. The eXTolBar.CRD component lets the user changes its visual appearance using skins, each one providing an additional visual experience that enhances viewing pleasure. The eXToolBar.CRD component it's free to use so no nag screens, no limitation, no evaluation message. Enjoy it.

Features include:

- Ability to layout the elements using the CRD format
- EBN support
- Built-in HTML support
- Multiple-lines HTML ToolTip support
- Icons/Pictures support
- Anchor / Link support
- and much more



Ž ExToolBar.CRD is a trademark of Exontrol. All Rights Reserved.

How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AppearanceEnum

The AppearanceEnum type specifies the way the toolbar's border is shown. The Appearance property retrieves or sets the control's appearance. The AppearanceEnum type supports the following values:

Name	Value	Description
TopBottom	0	A top and bottom border is shown.
Flat	1	A flat border is shown around the toolbar.
Sunken	2	A sunken border is shown around the toolbar.
Raised	3	A raised border is shown around the toolbar.
Etched	4	A etched border is shown around the toolbar.
Bump	5	A bump border is shown around the toolbar.

constants BackgroundPartEnum

The BackgroundPartEnum type indicates parts in the control. Use the Background property to specify a background color or a visual appearance for specific parts in the control. A Color expression that indicates the background color for a specified part. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part. The BackgroundPartEnum supports the following values:

Name	Value	Description
exToolTipAppearance	64	exToolTipAppearance. Specifies the visual appearance of the borders of the tooltips.
exToolTipBackColor	65	exToolTipBackColor. Specifies the tooltip's background color.
exToolTipForeColor	66	exToolTipForeColor. Specifies the tooltip's foreground color.
exToolBarButtonUpBackColor	151	exToolBarButtonUpBackColor. Specifies the visual appearance of the item when it is up.
exToolBarButtonUpForeColor	152	exToolBarButtonUpForeColor. Specifies the foreground color of the item when it is up.
exToolBarButtonDownBackColor	153	exToolBarButtonDownBackColor. Specifies the visual appearance of the item when it is down.
exToolBarButtonDownForeColor	154	exToolBarButtonUpForeColor. Specifies the foreground color of the item when it is down.
exToolBarButtonHotBackColor	155	exToolBarButtonHotBackColor. Specifies the visual appearance of the item when the cursor hovers it.
exToolBarButtonHotForeColor	156	exToolBarButtonHotForeColor. Specifies the foreground color of the item when the cursor hovers it.

constants PictureDisplayEnum

The PictureDisplayEnum type defines the way the control's Picture is arranged on the control. The Picture property assign a picture to be displayed on the control's background. The PictureDisplay property indicates how the picture is layout on the control's background. The PictureDisplayEnum type supports the following values:

Name	Value	Description
UpperLeft	0	The picture is vertically aligned at the top, and horizontally aligned on the left.
UpperCenter	1	The picture is vertically aligned at the top, and horizontally aligned at the center.
UpperRight	2	The picture is vertically aligned at the top, and horizontally aligned on the right.
MiddleLeft	16	The picture is vertically aligned in the middle, and horizontally aligned on the left.
MiddleCenter	17	The picture is vertically aligned in the middle, and horizontally aligned at the center.
MiddleRight	18	The picture is vertically aligned in the middle, and horizontally aligned on the right.
LowerLeft	32	The picture is vertically aligned at the bottom, and horizontally aligned on the left.
LowerCenter	33	The picture is vertically aligned at the bottom, and horizontally aligned at the center.
LowerRight	34	The picture is vertically aligned at the bottom, and horizontally aligned on the right.
Tile	48	Tiles the picture on the source.
Stretch	49	The picture is resized to fit the source.

constants StateEnum

For internal use only.

Name	Value	Description
exNormal	0	exNormal
exPushed	1	exPushed
exDisabled	2	exDisabled
exHover	3	exHover

Appearance object

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

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

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

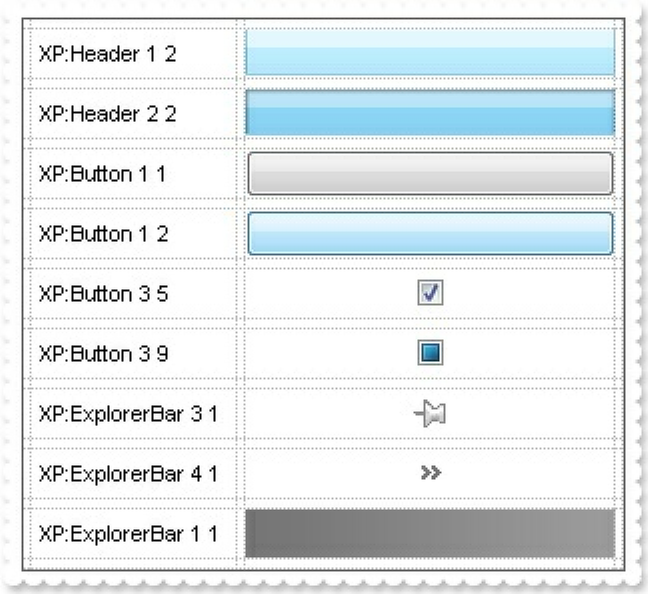
Adds or replaces a skin object to the control.

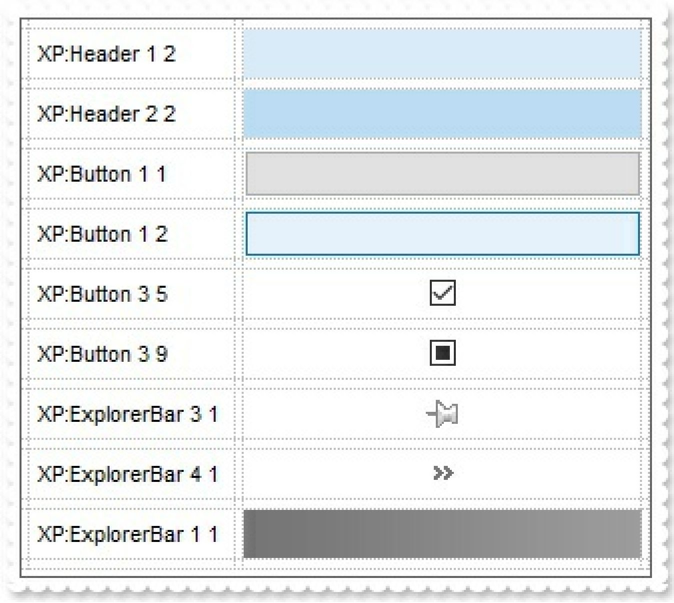
Type	Description
ID as Long	A Long expression that indicates the index of the skin being added or replaced. The value must be between 1 and 126, so Appearance collection should holds no more than 126 elements.
	The Skin parameter of the Add method can a STRING as explained bellow, a BYTE[] / safe arrays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

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

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

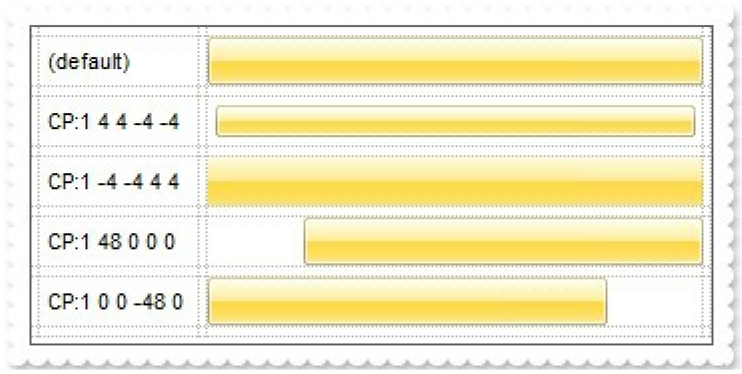
Skin as Variant





- A copy of another skin with different coordinates (position, size), if the Skin parameter starts with "**CP**:". Use this option, to display the EBN, using different coordinates (position, size). By default, the EBN skin object is rendered on the part's client area. Using this option, you can display the same EBN, on a different position / size. In this case, the syntax of the Skin parameter is: "CP:ID Left Top Right Bottom" where the ID is the identifier of the EBN to be used (it is a number that specifies the ID parameter of the Add method), Left, Top, Right and Bottom parameters/numbers specifies the relative position to the part's client area, where the EBN should be rendered. The Left, Top, Right and Bottom parameters are numbers (negative, zero or positive values, with no decimal), that can be followed by the D character which indicates the value according to the current DPI settings. For instance, "CP:1 -2 -2 2 2", uses the EBN with the identifier 1, and displays it on a 2-pixels wider rectangle no matter of the DPI settings, while "CP:1 -2D -2D 2D 2D" displays it on a 2-pixels wider rectangle if DPI settings is 100%, and on on a 3-pixels wider rectangle if DPI settings is 150%.

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



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

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control, when the "XP:" prefix is not specified in the Skin parameter (available for Windows XP systems). By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part. Use the Remove method to remove a specific skin from the control. Use the Clear method to remove all skins in the control.

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

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

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

property Appearance.RenderType as Long

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

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

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

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

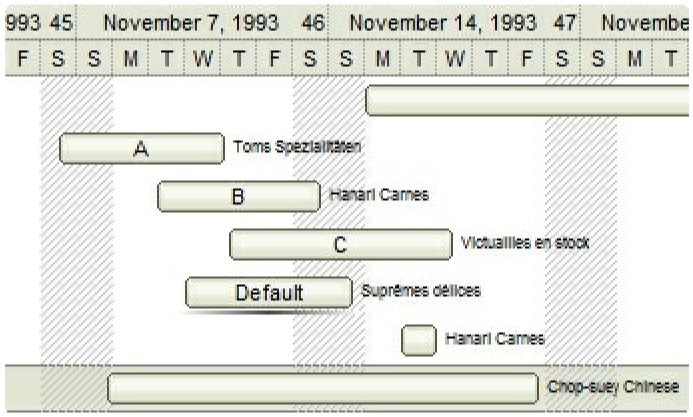
Click here ▶ to watch a movie on how you can change the colors to be applied on EBN objects.

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

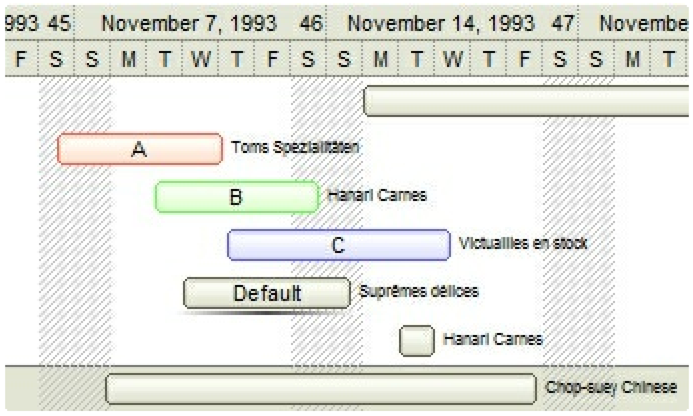
- "A" in Red (RGB(255,0,0), for instance the bar's property exBarColor is 0x10000FF
- "B" in Green (RGB(0,255,0), for instance the bar's property exBarColor is 0x100FF00
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance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

The RenderType property could be one of the following:

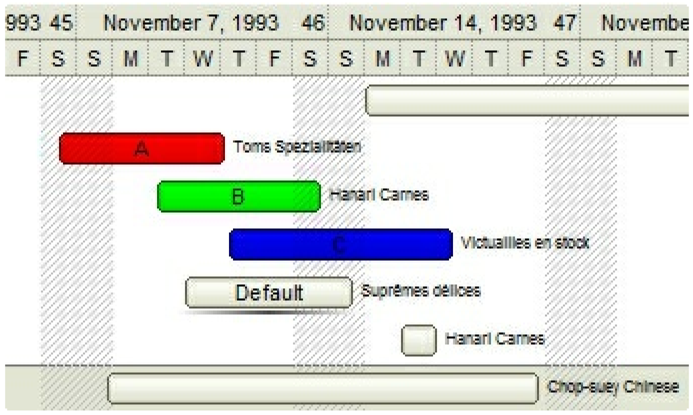
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



- ***-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), ...)

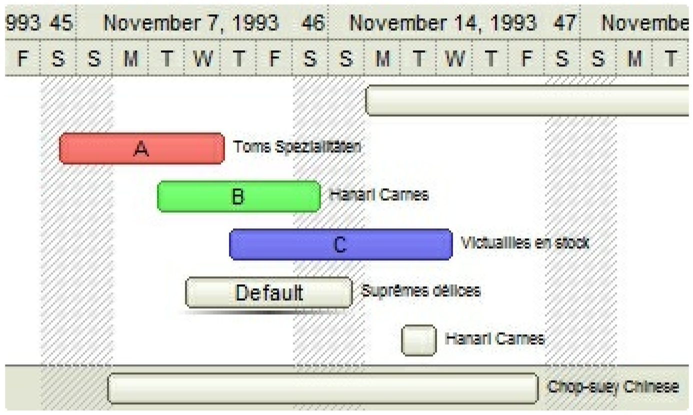


- ***-1**, AND-color scheme*, The color to be applied on the part of the control is an AND bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the AND bit for the entire Blue channel, or in other words, it applies a more Blue 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), ...)



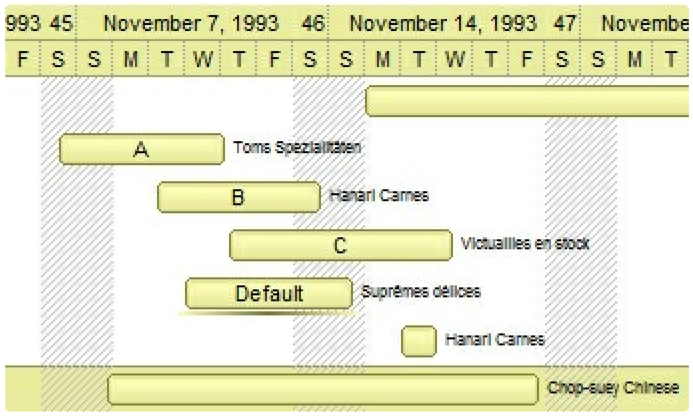
- ***0**, default*, the specified color is applied to the EBN. For instance, the

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.

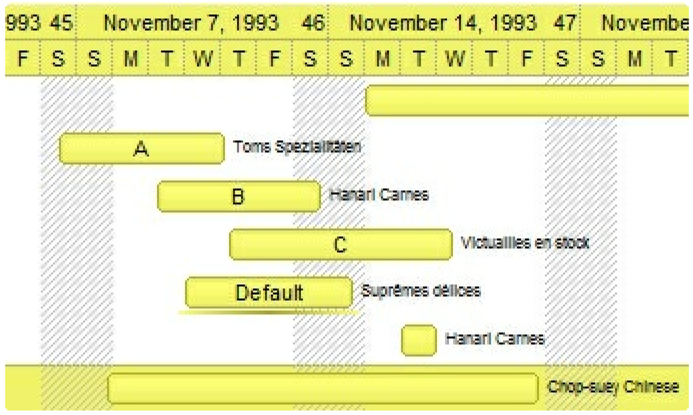


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on 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.

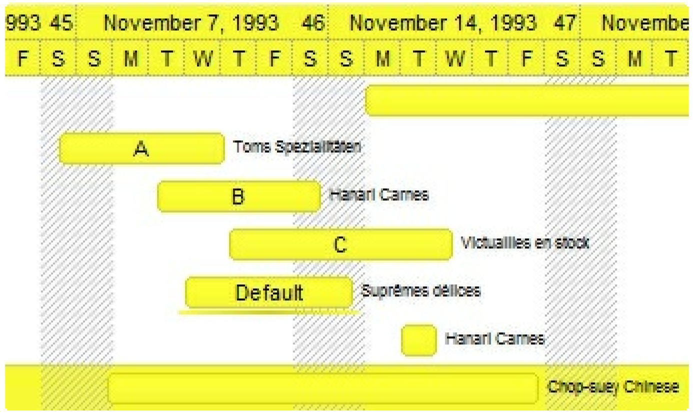
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



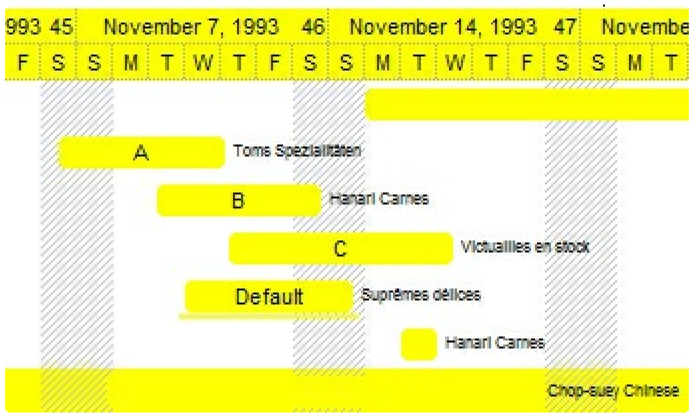
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



*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):*



Item object

The Item object holds information about an item to be shown on the toolbar. The Item property gives access to an item based on its identifier. The Format property specifies the CRD format to arrange the objects inside the control. The Item object supports the following properties and methods:

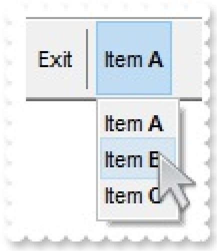
Name	Description
Caption	Specifies the item's HTML caption.
Enabled	Indicates if the item is enabled or disabled.
ID	Specifies the item's identifier.
ToolTip	Specifies the item's HTML tooltip.
UserData	Specifies the item's user data.

property Item.Caption as String

Specifies the item's HTML caption.

Type	Description
String	A String expression that defines the built-in HTML caption to be shown on the item's toolbar.

By default, the Caption property is empty. The Caption property specifies the item's HTML caption. If the Caption property includes the ItemsDelimiter value, it indicates that the toolbar's item displays a drop down list, when user clicks the item. Each part of the Caption between two delimiters indicates an item in the drop down. The ItemValueDelimiter property specifies the delimiter sequence for drop down value. For instance, "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3" indicates that the toolbar's item displays the "Item **A**", and it's drop down list contains "Item **A**" with the value 1, "Item **B**" with the value 2 and "Item **C**" with the value 3.



The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The UserData property associates any extra data to an item. The Enabled property enables or disables the specified item.

The Caption property supports the following HTML elements:

- **<b> ... </b>** displays the text in **bold**
- **<i> ... </i>** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... </a>** 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.
- **<font face;size> ... </font>** displays portions of text with a different font and/or different size. For instance, the "<font Tahoma;12>bit</font>" 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, "<font

;12>bit</font>" 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
- **<br>** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&amp;** (&), **&lt;** (<), **&gt;** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the &#8364; displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display <b>bold</b> in HTML caption you can use &lt;b&gt;bold&lt;/b&gt;
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the <font face;size> to define a smaller or a larger font

to be displayed. For instance: "Text with <font ;7><**off** 6>subscript" displays the text such as: Text with subscript The "Text with <font ;7><**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 <font> 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 "<font ;18><**gra** FFFFFF;1;1>gradient-center</**gra**></font>" 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 <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**></font>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**sha**>shadow</**sha**></font>" generates the following picture:

or "*<font ;31><**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**></font>*" gets:

outline anti-aliasing

property Item.Enabled as Boolean

Indicates if the item is enabled or disabled.

Type	Description
Boolean	A Boolean expression that specifies whether the item is enabled or disabled.

By default, the Enabled property is True. The Enabled property enables or disables the specified item. The Caption property specifies the item's HTML caption. The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The UserData property associates any extra data to an item.

property Item.ID as Long

Specifies the item's identifier.

Type	Description
Long	A Long expression that defines the item's identifier. The Format property specifies the CRD format to arrange the objects inside the control.

The ID property specifies the item's identifier (specified by the Format property). The UserData property associates any extra data to an item. The Caption property specifies the item's HTML caption. The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The Enabled property enables or disables the specified item.

property Item.ToolTip as String

Specifies the item's HTML tooltip.

Type	Description
String	A String expression that specifies the HTML text to be shown when the cursor hovers the item

By default, the ToolTip property is empty. The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The Caption property specifies the item's HTML caption. The UserData property associates any extra data to an item. The Enabled property enables or disables the specified item.

The Caption property supports the following HTML elements:

- **<b> ... </b>** displays the text in **bold**
- **<i> ... </i>** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... </a>** 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.
- **<font face;size> ... </font>** displays portions of text with a different font and/or different size. For instance, the "<font Tahoma;12>bit</font>" 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, "<font ;12>bit</font>" 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
- **<br>** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&amp;** (&), **&lt;** (<), **&gt;** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the &#8364; displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display <b>bold</b> in HTML caption you can use &lt;b&gt;bold&lt;/b&gt;
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the <font face;size> to define a smaller or a larger font to be displayed. For instance: "Text with <font ;7><**off** 6>subscript" displays the text such as: Text with subscript The "Text with <font ;7><**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 <font> 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 "<font ;18><**gra** FFFFFF;1;1>gradient-center</**gra**></font>" 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 <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**></font>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**sha**>shadow</**sha**></font>" generates the following picture:

or "*<font ;31><**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**></font>*" gets:

outline anti-aliasing

property Item.UserData as Variant

Specifies the item's user data.

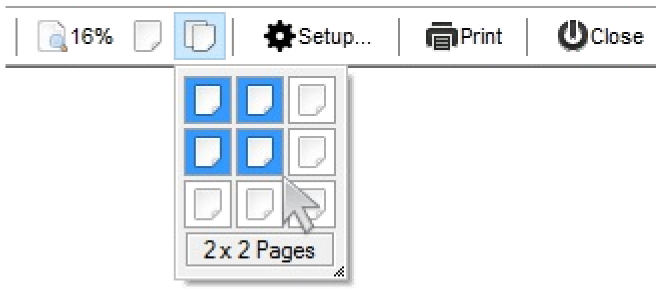
Type	Description
Variant	Any VARIANT expression that's associated with the current item

By default, the UserData property is empty. The UserData property associates any extra data to an item. The Caption property specifies the item's HTML caption. The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The Enabled property enables or disables the specified item.

ToolBarCRD 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: {DD586AE6-F2A0-4308-8F34-8016B16F000E}. The object's program identifier is: "Exontrol.ToolBar.CRD". The /COM object module is: "ExToolBar.CRD.dll"

The eXToolBar.CRD component is a graphical control element on which on-screen buttons, icons, menus, or other input or output elements are placed. The eXTolBar.CRD component lets the user changes its visual appearance using skins, each one providing an additional visual experience that enhances viewing pleasure. The eXToolBar.CRD component it's free to use so no nag screens, no limitation, no evaluation message. Enjoy it.



Use the Format property to add/remove items in the toolbar control. Use the Item property to access the Item object. The ToolBarCRD object supports the following properties and methods.

Name	Description
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
BackColor	Specifies the control's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
Debug	Specifies whether the control displays debug 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.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
Format	Specifies the CRD format to arrange the objects inside the control.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Images List Control.
ImageSize	Retrieves or sets the size of icons the control displays.
Item	Retrieves an item from the toolbar.
ItemFromPoint	Retrieves the index of the item from the point.
ItemsDelimiter	Specifies the delimiter sequence for drop down items.
ItemValueDelimiter	Specifies the delimiter sequence for drop down value.
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	Refreses the control.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowToolTip	Shows the specified tooltip at given position.
Template	Specifies the control's template.
TemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.
ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipFont	Retrieves or sets the tooltip's font.
ToolTipPopDelay	Specifies the period in ms of time the ToolTip remains

	visible if the mouse pointer is stationary within a control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.

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

Retrieves the identifier of the anchor from point.

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

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

property ToolBarCRD.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

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

By default, the Appearance property is, TopBottom. Use the Appearance property to specify the control's border.

method ToolBarCRD.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 ToolBarCRD1_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 ToolBarCRD.BackColor as Color

Specifies the control's background color.

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

Use the BackColor property to specify a solid color on the control's background. The ForeColor property specifies the control's foreground color. The Picture property to assign your logo on the control's background. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the Background(exToolBarButtonHotBackColor) property to specify the visual appearance of the item when the cursor hovers it.

property ToolBarCRD.Background(Part as BackgroundPartEnum) as Color

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

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

The Background property specifies a background color or a visual appearance for specific parts in the control. If the Background property is 0, the control draws the part as default. Use the Add method to add new skins to the control. Use the Remove method to remove a specific skin from the control. Use the Clear method to remove all skins in the control.

How can I change the background of the hovering item?

VBA (MS Access, Excell...)

```
With ToolBarCRD1
	.VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
	.Background(155) = &H1000000
	.Format = "1,2,3,4"
End With
```

VB6

```
With ToolBarCRD1
	.VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
	.Background(exToolBarButtonHotBackColor) = &H1000000
	.Format = "1,2,3,4"
End With
```

VB.NET

```
With Extoolbarcrd1
    .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")

.set_Background32(exontrol.EXTOOLBARCRDLib.BackgroundPartEnum.exToolBarButtor

    .Format = "1,2,3,4"
End With
```

VB.NET for /COM

```
With AxToolBarCRD1
    .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")

.set_Background(EXTOOLBARCRDLib.BackgroundPartEnum.exToolBarButtonHotBackC

    .Format = "1,2,3,4"
End With
```

C++

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

    #import <ExToolBar.CRD.dll>
    using namespace EXTOOLBARCRDLib;
*/
EXTOOLBARCRDLib::IToolBarCRDPtr spToolBarCRD1 =
GetDlgItem(IDC_TOOLBARCRD1)->GetControlUnknown();
spToolBarCRD1->GetVisualAppearance()-
>Add(1,"c:\\exontrol\\images\\normal.ebn");
spToolBarCRD1-
>PutBackground(EXTOOLBARCRDLib::exToolBarButtonHotBackColor,0x1000000);
spToolBarCRD1->PutFormat(L"1,2,3,4");
```

C++ Builder

```
ToolBarCRD1->VisualAppearance-
>Add(1,TVariant("c:\\exontrol\\images\\normal.ebn"));
ToolBarCRD1-
>Background[Extoolbarcrdlib_tlb::BackgroundPartEnum::exToolBarButtonHotBackCol
 = 0x1000000;
ToolBarCRD1->Format = L"1,2,3,4";
```

C#

```
extoolbarcrd1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
extoolbarcrd1.set_Background32(exontrol.EXTOOLBARCRDLib.BackgroundPartEnum.ex

extoolbarcrd1.Format = "1,2,3,4";
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:DD586AE6-F2A0-4308-8F34-8016B16F000E"
id="ToolBarCRD1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
    ToolBarCRD1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
    ToolBarCRD1.Background(155) = 16777216;
    ToolBarCRD1.Format = "1,2,3,4";
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:DD586AE6-F2A0-4308-8F34-8016B16F000E"
id="ToolBarCRD1"></OBJECT>
```

```
<SCRIPT LANGUAGE="VBScript">
Function Init()
    With ToolBarCRD1
        .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
        .Background(155) = &H1000000
        .Format = "1,2,3,4"
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axToolBarCRD1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
axToolBarCRD1.set_Background(EXTOOLBARCRDLib.BackgroundPartEnum.exToolBarB

axToolBarCRD1.Format = "1,2,3,4";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    ;

    super();

    extoolbarcrd1.VisualAppearance().Add(1,"c:\\exontrol\\images\\normal.ebn");
    extoolbarcrd1.Background(155/*exToolBarButtonHotBackColor*/,0x1000000);
    extoolbarcrd1.Format("1,2,3,4");
}
```

Delphi 8 (.NET only)

```
with AxToolBarCRD1 do
begin
```

```
    VisualAppearance.Add(1,'c:\exontrol\images\normal.ebn');

set_Background(EXTOOLBARCRDLib.BackgroundPartEnum.exToolBarButtonHotBackC

    Format := '1,2,3,4';
end
```

Delphi (standard)

```
with ToolBarCRD1 do
begin
    VisualAppearance.Add(1,'c:\exontrol\images\normal.ebn');
    Background[EXTOOLBARCRDLib_TLB.exToolBarButtonHotBackColor] := $1000000;
    Format := '1,2,3,4';
end
```

VFP

```
with thisform.ToolBarCRD1
    .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
    .Object.Background(155) = 0x1000000
    .Format = "1,2,3,4"
endwith
```

dBASE Plus

```
local oToolBarCRD

oToolBarCRD = form.EXTOOLBAR_CRDACTIVEXCONTROL1.nativeObject
oToolBarCRD.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
oToolBarCRD.Template = [Background(155) = 16777216] // 
oToolBarCRD.Background(155) = 0x1000000
oToolBarCRD.Format = "1,2,3,4"
```

XBasic (Alpha Five)

```
Dim oToolBarCRD as P
```

```
oToolBarCRD = topparent:CONTROL_ACTIVEX1.activex
oToolBarCRD.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
oToolBarCRD.Template = "Background(155) = 16777216" //
oToolBarCRD.Background(155) = 16777216
oToolBarCRD.Format = "1,2,3,4"
```

Visual Objects

```
oDCOCX_Exontrol1:VisualAppearance:Add(1,"c:\exontrol\images\normal.ebn")
oDCOCX_Exontrol1:[Background,exToolBarButtonHotBackColor] := 0x1000000
oDCOCX_Exontrol1:Format := "1,2,3,4"
```

PowerBuilder

```
OleObject oToolBarCRD

oToolBarCRD = ole_1.Object
oToolBarCRD.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
oToolBarCRD.Background(155,16777216 /*0x1000000*/)
oToolBarCRD.Format = "1,2,3,4"
```

Visual DataFlex

```
Procedure OnCreate
  Forward Send OnCreate
  Variant voAppearance
  Get ComVisualAppearance to voAppearance
  Handle hoAppearance
  Get Create (RefClass(cComAppearance)) to hoAppearance
  Set pvComObject of hoAppearance to voAppearance
    Get ComAdd of hoAppearance 1 "c:\exontrol\images\normal.ebn" to Nothing
  Send Destroy to hoAppearance
  Set ComBackground OLEexToolBarButtonHotBackColor to |CI$1000000
  Set ComFormat to "1,2,3,4"
```

```
End_Procedure
```

XBase++

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

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

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

   oToolBarCRD := XbpActiveXControl():new( oForm:drawingArea )
   oToolBarCRD:CLSID  := "Exontrol.ToolBar.CRD.1" /*{DD586AE6-F2A0-4308-8F34-
8016B16F000E}*/
   oToolBarCRD:create(,, {10,60},{610,370} )

      oToolBarCRD:VisualAppearance():Add(1,"c:\exontrol\images\normal.ebn")

oToolBarCRD:SetProperty("Background",155/*exToolBarButtonHotBackColor*/,0x10000

      oToolBarCRD:Format := "1,2,3,4"

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

method ToolBarCRD.BeginUpdate ()

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

Type	Description

This method prevents the control from painting until the EndUpdate method is called. Use the Refresh method to refresh the control

property ToolBarCRD.Debug as Boolean

Specifies whether the control displays debug information.

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

By default, the Debug property is False. The Debug property can be used to display the item's identifiers at runtime.

property ToolBarCRD.Enabled as Boolean

Enables or disables the control.

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

By default, the Enabled property is True. The Enabled property enables or disables the control. The Enabled property enables or disables the specified item. The Caption property specifies the item's HTML caption. The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The UserData property associates any extra data to an item.

method ToolBarCRD.EndUpdate ()

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

Type	Description

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

property ToolBarCRD.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 ToolBarCRD.ExecuteTemplate (Template as String)

Executes a template and returns the result.

Type	Description
Template as String	A Template string being executed
Return	**Description**
Variant	A Variant expression that indicates the result after executing the Template.

Use the ExecuteTemplate property to returns the result of executing a template file. Use the Template property to execute a template without returning any result. Use the ExecuteTemplate property to execute code by passing instructions as a string (template string). For instance, you can use the EXPRINT.PrintExt = CONTROL.ExecuteTemplate("me") to print the control's content.

For instance, the following sample retrieves the the handle of the first visible item:

```
Debug.Print ToolBarCRD1.ExecuteTemplate("Items.FirstVisibleItem()")
```

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 ToolBarCRD.Font as IFontDisp

Retrieves or sets the control's font.

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

Use the Font property to change the control's font . Use the Refresh method to refresh the control. Use the BeginUpdate and EndUpdate method to maintain performance while adding new columns or items.

property ToolBarCRD.ForeColor as Color

Specifies the control's foreground color.

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

The ForeColor property specifies the control's foreground color. Use the BackColor property to specify a solid color on the control's background. The Picture property to assign your logo on the control's background. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the Background(exToolBarButtonHotForeColor) property to specify the visual appearance of the item when the cursor hovers it.

property ToolBarCRD.Format as String

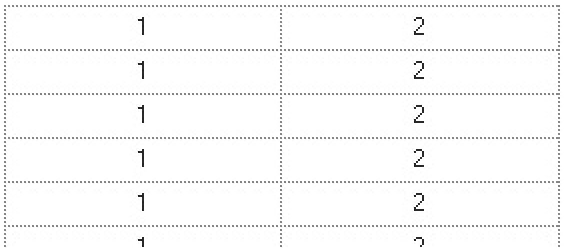
Specifies the CRD format to arrange the objects inside the control.

Type	Description
String	A String expression that specifies CRD format to arrange the items on the toolbar. The Exontrol's Custom Row Designer (**exCRD**) is a WYSWYG tool to build new layouts for cells/nodes, items/rows or columns/fields. The exCRD tool generates CRD strings from the layout you built. The syntax of CRD strings is designed to be easy to build, change and read. Using CRD strings is powerful than preformatted card view, group view formats, nested bands, and so on, since you are free to define the full layout of the cell/node, item/row or a column/field.

The Format property adds/removes items in the toolbar control. The Item property retrieves an item from the toolbar. The Caption property specifies the item's HTML caption. The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The UserData property associates any extra data to an item. The Enabled property enables or disables the specified item. The Select event notifies your application once the user clicks / selects the item.

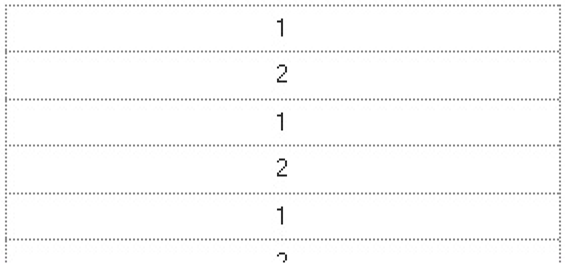
For instance, here are few simple CRD strings:

- The CRD string `**1,2**` divides the cell in two parts, the left side displays the first column, and the right part displays the second column. Similar with horizontally splitting a cell in two pieces.

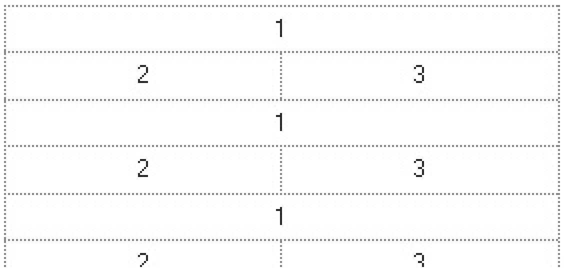


1	2
1	2
1	2
1	2
1	2

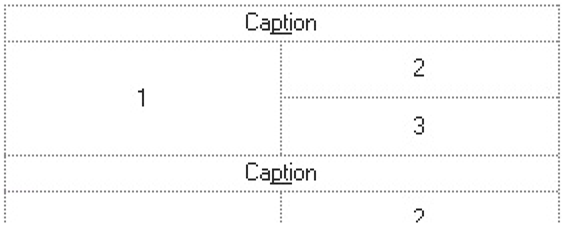
- The CRD string `**1/2**` splits vertically the cell in two parts, where the upper part displays the first column, and the down part displays the second column. Similar with vertically splitting a cell in two pieces.



- The CRD string `**1/2,3**` splits a cell in two, the upper part displays the first column, the bottom part is divided in other two parts, where the left part displays the second column, and the right part displays the third column.



- The CRD string `**18;"Ca<u>pti</u>on"[a=17]/1,(2/3)**` splits vertically the cell in two parts, the upper part displays the "Caption" string aligned on the center, with the height of 18 pixels, the bottom part is divided in other two parts, the left part displays the first column, and the second part is vertically divided in other two parts, where the upper part displays the second column and the bottom part displays the third column.



The CRD syntax in BNF notation is defined like follows:

```
<CRD> ::= [<Options>] <GroupCRD>
<GroupCRD> ::= <UpPart> [ "|" <DownPart> ]
<UpPart> ::= <Lines>
<DownPart> ::= <Lines>
<Lines> ::= <Line> | "(" <Lines> ")" | <Lines> "/" <Lines>
<Line> ::= [<Height>;] <LeftPart> [ "|" <RightPart> ]
<LeftPart> ::= <Fields>
<RightPart> ::= <Fields>
<Fields> ::= <Field> | "(" <Fields> ")" | <Fields> "," <Fields>
<Field> ::= <Identifier> [<Options>] [ ":" <Width>]
```

```
<Identifier> ::= <Index> | <Caption>
<Options> ::= <Options> [ "[" <Option> "]" ]
<Option> ::= <Property> [ "=" <Value> ]
<Property> ::= <letter> | <Property> [ <letter> | <digit> ]
<Value> ::= <Number> | <String>
<Index> ::= <Number>
<Caption> ::= <String>
<Width> ::= <Number>
<Height> ::= <Number>
<Number> ::= <digit><Number>
<String> ::= """<any_character>"""
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

The CRD strings may include the following elements:

- **Index**, represents numbers, a set of digits. For instance, 1 2 3, ... and so on
- **Caption**, represents strings, and are delimitated by " characters. For instance, "" (empty string) "<b>Caption</b>" "test", and so on
- **,** (**field separator**), delimits the elements in the same line. For instance: 1,"Caption",(3/4/5)
- **/**, (**line separator**), delimits the elements in different lines. For instance: 1/"Caption"/(2,3,4)
- **|**, (**divider character**), splits the left and right parts of a line, or top or bottom parts of a group
- **()**, (**groups**), defines a group
- **[]**, (**options**), specifies options for elements in the layout
- **;**, (**line's height separator**), specifies the height of the line or the group
- **:**, (**field's width separator**), specifies the width of the field or the group

The **Index** and **Caption** element may have one or more of the following options:

- **Border** option, [b=<Number>], specifies which borders are shown or hidden. The <Number> may be a sum of one or more values like follows:

  - **1**, top border, draws the top border
  - **2**, right border, draws the right border
  - **4**, bottom border, draws the bottom border
  - **8**, left border, draws the left border

  For instance, the [b=5] means that the element draws the top and the bottom borders. For instance, if the [b=0] is at the beginning of the CRD string, it specifies that by default, no borders are shown.

- **Background** option, [bg=RGB(,<Number>,<Number>,<Number>)] | [bg=<Number>], specifies the background color of the element.
- **Foreground** option, [fg=RGB(,<Number>,<Number>,<Number>)] | [fg=<Number>], specifies the foreground color of the element. This option has effect only for Caption elements.

The **Caption** element may have one or more of the following options:

- **Alignment** option, [a=<Number>], specifies the alignment of the caption in the element. By default, if the option is missing, the caption is aligned to the left. The <Number> may be one of the values like follows:

    - **0**, TopLeft, Aligns the caption to the top left corner.
    - **1**, TopCenter, Centers the caption on the top edge.
    - **2**, TopRight, Aligns the caption to the upper right corner.
    - **16**, MiddleLeft, Aligns horizontally the caption on the left side, and centers the caption vertically
    - **17**, MiddleCenter, Puts the caption on the center of the element. (Default)
    - **18**, MiddleRight, Aligns horizontally the caption on the right side, and centers the caption vertically
    - **32**, BottomLeft, Aligns the caption to the lower left corner
    - **33**, BottomCenter, Centers the caption on the lower edge
    - **34**, BottomLeft, The caption is resized to fit the source

- **WordWrap** option, [ww], specifies whether the caption is wrapping in the element's client area. If the option is present, the text is arranged on multiple lines, else the text is displayed on a single line.

The options for the CRD string may be (these options must be always at the beginning of the CRD string):

- **Debug** option, [debug], displays debug information when running in the component. Has no effect in the exCRD tool
- **Border** option, [b=<Number>], specifies which borders are shown or hidden, for all elements in the CRD string. The <Number> may be a sum of one or more values like follows:

    - **1**, top border, all elements in the CRD layout draw the top border
    - **2**, right border, all elements in the CRD layout draw the right border
    - **4**, bottom border, all elements in the CRD layout draw the bottom border
    - **8**, left border, all elements in the CRD layout draw the left border

- **DrawGridLines** options, [dgl=0|1|-1], specifies whether the CRD layout draws the grid lines. This option is depending on the component's context.

property ToolBarCRD.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

Type	Description
New as Boolean	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 ToolBarCRD.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.

By default, the HTMLPicture collection is empty. The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> tags. Use the HTMLPicture property to add new pictures to be used in HTML captions. For instance, the HTMLPicture("pic1") = "c:\winnt\zapotec.bmp", loads the zapotec picture and associates the pic1 key to it. Any "<img>pic1</img>" sequence in HTML captions, displays the pic1 picture. On return, the HTMLPicture property retrieves a Picture object (this implements the IPictureDisp interface). The Images method specifies the list of 16x16 icons to be displayed on the control's surface. The Caption property specifies the caption of the item (including icons, picture and so on).

property ToolBarCRD.hWnd as Long

Retrieves the control's window handle.

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

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

method ToolBarCRD.Images (Handle as Variant)

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

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

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

The Images method assigns a list of icons to be displayed on the control's surface. The icons can be displayed on the control's using the <img>number</img> HTML tags. The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img>key</img> tags. The ReplaceIcon method replaces icons in the control's . The Caption property specifies the caption of the item (including icons, picture and so on).

property ToolBarCRD.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 ToolBarCRD.Item (Index as Variant) as Item

Retrieves an item from the toolbar.

Type	Description
Index as Variant	A Long expression that specifies the identifier of the Item to be requested.
Item	An Item object being requested.

The Item property retrieves an item from the toolbar. The Format property adds/removes items in the toolbar control. The Caption property specifies the item's HTML caption. The ToolTip property specifies the item's tooltip which is shown when the cursor hovers it. The UserData property associates any extra data to an item. The Enabled property enables or disables the specified item. The Select event notifies your application once the user clicks / selects the item.

How can I display a drop-down panel?

VBA (MS Access, Excell...)

```
' Select event - Notifies once the user clicks the item.
Private Sub ToolBarCRD1_Select(ByVal ID As Variant,ByVal SelectedID As Variant)
	With ToolBarCRD1
		Debug.Print( "Select" )
		Debug.Print( SelectedID )
	End With
End Sub

With ToolBarCRD1
	.ItemsDelimiter = ";"
	.Format = "1,-1,2"
	.Item(1).Caption = "Exit"
	.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

VB6

```
' Select event - Notifies once the user clicks the item.
Private Sub ToolBarCRD1_Select(ByVal ID As Variant,ByVal SelectedID As Variant)
	With ToolBarCRD1
```

```
        Debug.Print( "Select" )
        Debug.Print( SelectedID )
    End With
End Sub

With ToolBarCRD1
    .ItemsDelimiter = ";"
    .Format = "1,-1,2"
    .Item(1).Caption = "Exit"
    .Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

VB.NET

```
' Select event - Notifies once the user clicks the item.
Private Sub Extoolbarcrd1_Select(ByVal sender As System.Object,ByVal ID As
Object,ByVal SelectedID As Object) Handles Extoolbarcrd1.Select
    With Extoolbarcrd1
        Debug.Print( "Select" )
        Debug.Print( SelectedID )
    End With
End Sub

With Extoolbarcrd1
    .ItemsDelimiter = ";"
    .Format = "1,-1,2"
    .Item(1).Caption = "Exit"
    .Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

VB.NET for /COM

```
' Select event - Notifies once the user clicks the item.
Private Sub AxToolBarCRD1_Select(ByVal sender As System.Object, ByVal e As
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent) Handles
AxToolBarCRD1.Select
    With AxToolBarCRD1
        Debug.Print( "Select" )
        Debug.Print( e.selectedID )
    End With
End Sub

With AxToolBarCRD1
    .ItemsDelimiter = ";"
    .Format = "1,-1,2"
    .get_Item(1).Caption = "Exit"
    .get_Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

C++

```
// Select event - Notifies once the user clicks the item.
void OnSelectToolBarCRD1(VARIANT   ID,VARIANT   SelectedID)
{
    /*
        Copy and paste the following directives to your header file as
        it defines the namespace 'EXTOOLBARCRDLib' for the library: 'ExToolBar.CRD
1.0 Control Library'
        #import <ExToolBar.CRD.dll>
        using namespace EXTOOLBARCRDLib;
     */
    EXTOOLBARCRDLib::IToolBarCRDPtr spToolBarCRD1 =
GetDlgItem(IDC_TOOLBARCRD1)->GetControlUnknown();
    OutputDebugStringW( L"Select" );
    OutputDebugStringW( L"SelectedID" );
}

EXTOOLBARCRDLib::IToolBarCRDPtr spToolBarCRD1 =
GetDlgItem(IDC_TOOLBARCRD1)->GetControlUnknown();
spToolBarCRD1->PutItemsDelimiter(L";");
spToolBarCRD1->PutFormat(L"1,-1,2");
spToolBarCRD1->GetItem(long(1))->PutCaption(L"Exit");
spToolBarCRD1->GetItem(long(2))->PutCaption(L"Item <b>A;Item <b>A#1;Item
<b>B#2;Item <b>C#3");
```

C++ Builder

```
// Select event - Notifies once the user clicks the item.
void __fastcall TForm1::ToolBarCRD1Select(TObject *Sender,Variant   ID,Variant
SelectedID)
{
    OutputDebugString( L"Select" );
    OutputDebugString( L"SelectedID" );
}

ToolBarCRD1->ItemsDelimiter = L";";
ToolBarCRD1->Format = L"1,-1,2";
ToolBarCRD1->Item[TVariant(1)]->Caption = L"Exit";
ToolBarCRD1->Item[TVariant(2)]->Caption = L"Item <b>A;Item <b>A#1;Item
<b>B#2;Item <b>C#3";
```

C#

```
// Select event - Notifies once the user clicks the item.
private void extoolbarcrd1_Select(object sender,object   ID,object   SelectedID)
{
    System.Diagnostics.Debug.Print( "Select" );
    System.Diagnostics.Debug.Print( SelectedID.ToString() );
}
//this.extoolbarcrd1.Select += new
exontrol.EXTOOLBARCRDLib.exg2antt.SelectEventHandler(this.extoolbarcrd1_Select);

extoolbarcrd1.ItemsDelimiter = ";";
extoolbarcrd1.Format = "1,-1,2";
extoolbarcrd1[1].Caption = "Exit";
extoolbarcrd1[2].Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3";
```

JScript/JavaScript

```
<BODY onload="Init()">
<SCRIPT FOR="ToolBarCRD1" EVENT="Select(ID,SelectedID)" LANGUAGE="JScript">
```

```
    alert( "Select" );
    alert( SelectedID );
</SCRIPT>

<OBJECT CLASSID="clsid:DD586AE6-F2A0-4308-8F34-8016B16F000E"
id="ToolBarCRD1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
    ToolBarCRD1.ItemsDelimiter = ";";
    ToolBarCRD1.Format = "1,-1,2";
    ToolBarCRD1.Item(1).Caption = "Exit";
    ToolBarCRD1.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3";
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<SCRIPT LANGUAGE="VBScript">
Function ToolBarCRD1_Select(ID,SelectedID)
    With ToolBarCRD1
        alert( "Select" )
        alert( SelectedID )
    End With
End Function
</SCRIPT>

<OBJECT CLASSID="clsid:DD586AE6-F2A0-4308-8F34-8016B16F000E"
id="ToolBarCRD1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
```

```
    With ToolBarCRD1
        .ItemsDelimiter = ";"
        .Format = "1,-1,2"
        .Item(1).Caption = "Exit"
        .Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
// Select event - Notifies once the user clicks the item.
private void axToolBarCRD1_Select(object sender,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent e)
{
    System.Diagnostics.Debug.Print( "Select" );
    System.Diagnostics.Debug.Print( e.selectedID.ToString() );
}
//this.axToolBarCRD1.Select += new
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEventHandler(this.axToolBarCRD1_Sele


axToolBarCRD1.ItemsDelimiter = ";";
axToolBarCRD1.Format = "1,-1,2";
axToolBarCRD1[1].Caption = "Exit";
axToolBarCRD1[2].Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3";
```

X++ (Dynamics Ax 2009)

```
// Select event - Notifies once the user clicks the item.
void onEvent_Select(COMVariant   _ID,COMVariant   _SelectedID)
{
    ;
    print( "Select" );
    print( _SelectedID );
}

public void init()
{
    COM com_Item;
    anytype var_Item;
    ;

    super();

    extoolbarcrd1.ItemsDelimiter(";");
    extoolbarcrd1.Format("1,-1,2");
    var_Item =
COM::createFromObject(extoolbarcrd1.Item(COMVariant::createFromInt(1)));
com_Item = var_Item;
    com_Item.Caption("Exit");
    var_Item =
COM::createFromObject(extoolbarcrd1.Item(COMVariant::createFromInt(2)));
com_Item = var_Item;
    com_Item.Caption("Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3");
}
```

Delphi 8 (.NET only)

```
// Select event - Notifies once the user clicks the item.
procedure TWinForm1.AxToolBarCRD1_Select(sender: System.Object; e:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent);
begin
    with AxToolBarCRD1 do
    begin
        OutputDebugString( 'Select' );
        OutputDebugString( e.selectedID );
    end
end;

with AxToolBarCRD1 do
begin
```

```
    ItemsDelimiter := ';';
    Format := '1,-1,2';
    get_Item(TObject(1)).Caption := 'Exit';
    get_Item(TObject(2)).Caption := 'Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3';
end
```

Delphi (standard)

```
// Select event - Notifies once the user clicks the item.
procedure TForm1.ToolBarCRD1Select(ASender: TObject; ID : OleVariant;SelectedID :
OleVariant);
begin
    with ToolBarCRD1 do
    begin
        OutputDebugString( 'Select' );
        OutputDebugString( SelectedID );
    end
end;

with ToolBarCRD1 do
begin
    ItemsDelimiter := ';';
    Format := '1,-1,2';
    Item[OleVariant(1)].Caption := 'Exit';
    Item[OleVariant(2)].Caption := 'Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3';
end
```

VFP

```
*** Select event - Notifies once the user clicks the item. ***
LPARAMETERS ID,SelectedID
    with thisform.ToolBarCRD1
        DEBUGOUT( "Select" )
        DEBUGOUT( SelectedID )
    endwith
```

```
with thisform.ToolBarCRD1
	.ItemsDelimiter = ";"
	.Format = "1,-1,2"
	.Item(1).Caption = "Exit"
	.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
endwith
```

dBASE Plus

```
/*
with (this.EXTOOLBAR_CRDACTIVEXCONTROL1.nativeObject)
	Select = class::nativeObject_Select
endwith
*/
// Notifies once the user clicks the item.
function nativeObject_Select(ID,SelectedID)
	oToolBarCRD = form.EXTOOLBAR_CRDACTIVEXCONTROL1.nativeObject
	? "Select"
	? Str(SelectedID)
return

local oToolBarCRD,var_Item,var_Item1

oToolBarCRD = form.EXTOOLBAR_CRDACTIVEXCONTROL1.nativeObject
oToolBarCRD.ItemsDelimiter = ";"
oToolBarCRD.Format = "1,-1,2"
// oToolBarCRD.Item(1).Caption = "Exit"
var_Item = oToolBarCRD.Item(1)
with (oToolBarCRD)
	TemplateDef = [dim var_Item]
	TemplateDef = var_Item
	Template = [var_Item.Caption = "Exit"]
endwith
// oToolBarCRD.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
var_Item1 = oToolBarCRD.Item(2)
with (oToolBarCRD)
```

```
    TemplateDef = [dim var_Item1]
    TemplateDef = var_Item1
    Template = [var_Item1.Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"]
endwith
```

XBasic (Alpha Five)

```
' Notifies once the user clicks the item.
function Select as v (ID  as  A,SelectedID  as  A)
    oToolBarCRD = topparent:CONTROL_ACTIVEX1.activex
    ? "Select"
    ? SelectedID
end function

Dim oToolBarCRD as P
Dim var_Item as local
Dim var_Item1 as local

oToolBarCRD = topparent:CONTROL_ACTIVEX1.activex
oToolBarCRD.ItemsDelimiter = ";"
oToolBarCRD.Format = "1,-1,2"
' oToolBarCRD.Item(1).Caption = "Exit"
var_Item = oToolBarCRD.Item(1)
oToolBarCRD.TemplateDef = "dim var_Item"
oToolBarCRD.TemplateDef = var_Item
oToolBarCRD.Template = "var_Item.Caption = `Exit`"

' oToolBarCRD.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
var_Item1 = oToolBarCRD.Item(2)
oToolBarCRD.TemplateDef = "dim var_Item1"
oToolBarCRD.TemplateDef = var_Item1
oToolBarCRD.Template = "var_Item1.Caption = `Item <b>A;Item <b>A#1;Item
<b>B#2;Item <b>C#3`"
```

Visual Objects

```
METHOD OCX_Exontrol1Select(ID,SelectedID) CLASS MainDialog
   // Select event - Notifies once the user clicks the item.
   OutputDebugString(String2Psz( "Select" ))
   OutputDebugString(String2Psz( AsString(SelectedID) ))
RETURN NIL


oDCOCX_Exontrol1:ItemsDelimiter := ";"
oDCOCX_Exontrol1:Format := "1,-1,2"
oDCOCX_Exontrol1:[Item,1]:Caption := "Exit"
oDCOCX_Exontrol1:[Item,2]:Caption := "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
```

PowerBuilder

```
/*begin event Select(any  ID,any  SelectedID) - Notifies once the user clicks the item.*/
/*
    oToolBarCRD = ole_1.Object
    MessageBox("Information",string( "Select" ))
    MessageBox("Information",string( String(SelectedID) ))
*/
/*end event Select*/

OleObject oToolBarCRD

oToolBarCRD = ole_1.Object
oToolBarCRD.ItemsDelimiter = ";"
oToolBarCRD.Format = "1,-1,2"
oToolBarCRD.Item(1).Caption = "Exit"
oToolBarCRD.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
```

Visual DataFlex

```
// Notifies once the user clicks the item.
Procedure OnComSelect Variant   llID Variant   llSelectedID
    Forward Send OnComSelect llID llSelectedID
    Showln "Select" llSelectedID
End_Procedure

Procedure OnCreate
    Forward Send OnCreate
    Set ComItemsDelimiter to ";"
    Set ComFormat to "1,-1,2"
    Variant voItem
    Get ComItem 1 to voItem
    Handle hoItem
    Get Create (RefClass(cComItem)) to hoItem
    Set pvComObject of hoItem to voItem
        Set ComCaption of hoItem to "Exit"
    Send Destroy to hoItem
    Variant voItem1
    Get ComItem 2 to voItem1
    Handle hoItem1
    Get Create (RefClass(cComItem)) to hoItem1
    Set pvComObject of hoItem1 to voItem1
        Set ComCaption of hoItem1 to "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
    Send Destroy to hoItem1
End_Procedure
```

XBase++

```
PROCEDURE OnSelect(oToolBarCRD,ID,SelectedID)
    DevOut( "Select" )
    DevOut( Transform(SelectedID,"") )
RETURN

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

```
PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oToolBarCRD

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

   oToolBarCRD := XbpActiveXControl():new( oForm:drawingArea )
   oToolBarCRD:CLSID  := "Exontrol.ToolBar.CRD.1" /*{DD586AE6-F2A0-4308-8F34-
8016B16F000E}*/
   oToolBarCRD:create(,, {10,60},{610,370} )

      oToolBarCRD:Select := {|ID,SelectedID| OnSelect(oToolBarCRD,ID,SelectedID)}
/*Notifies once the user clicks the item.*/

      oToolBarCRD:ItemsDelimiter := ";"
      oToolBarCRD:Format := "1,-1,2"
      oToolBarCRD:Item(1):Caption := "Exit"
      oToolBarCRD:Item(2):Caption := "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"

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

property ToolBarCRD.ItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

Retrieves the index of the item from the point.

Type	Description
X as OLE_XPOS_PIXELS	A Long expression that specifies the x-cursor position.
Y as OLE_YPOS_PIXELS	A Long expression that specifies the y-cursor position.
Long	A Long expression that specifies the index of the item from the cursor or 0 if not found.

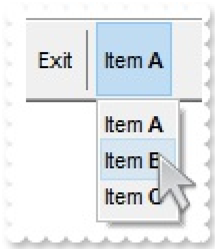
Use the ItemFromPoint property to get the item from cursor. The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. The MouseIn event occurs when the mouse enters the item. The MouseOut event occurs when the mouse exits the item. Use the Background(exToolBarButtonHotBackColor) property to specify the visual appearance of the item when the cursor hovers it.

property ToolBarCRD.ItemsDelimiter as String

Specifies the delimiter sequence for drop down items.

Type	Description
String	A String expression that specifies the delimiter of items in the Caption property to specify the items into a drop down item.

By default, the ItemsDelimiter property is "\r\n" (Chr(13) & Chr(10), Carriage return-linefeed combination). The ItemsDelimiter property specifies the delimiter sequence for drop down items. The ItemValueDelimiter property specifies the delimiter sequence for drop down value. The Caption property specifies the caption to be shown on the item. If the Caption property includes the ItemsDelimiter value, it indicates that the toolbar's item displays a drop down list, when user clicks the item. Each part of the Caption between two delimiters indicates an item in the drop down. For instance, "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3" indicates that the toolbar's item displays the "Item **A**", and it's drop down list contains "Item **A**" with the value 1, "Item **B**" with the value 2 and "Item **C**" with the value 3.



How can I display a drop-down panel?

VBA (MS Access, Excell...)

```
' Select event - Notifies once the user clicks the item.
Private Sub ToolBarCRD1_Select(ByVal ID As Variant,ByVal SelectedID As Variant)
    With ToolBarCRD1
        Debug.Print( "Select" )
        Debug.Print( SelectedID )
    End With
End Sub

With ToolBarCRD1
    .ItemsDelimiter = ";"
    .Format = "1,-1,2"
```

```
	.Item(1).Caption = "Exit"
	.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

VB6

```
' Select event - Notifies once the user clicks the item.
Private Sub ToolBarCRD1_Select(ByVal ID As Variant,ByVal SelectedID As Variant)
	With ToolBarCRD1
		Debug.Print( "Select" )
		Debug.Print( SelectedID )
	End With
End Sub

With ToolBarCRD1
	.ItemsDelimiter = ";"
	.Format = "1,-1,2"
	.Item(1).Caption = "Exit"
	.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

VB.NET

```
' Select event - Notifies once the user clicks the item.
Private Sub Extoolbarcrd1_Select(ByVal sender As System.Object,ByVal ID As
Object,ByVal SelectedID As Object) Handles Extoolbarcrd1.Select
	With Extoolbarcrd1
		Debug.Print( "Select" )
		Debug.Print( SelectedID )
	End With
End Sub

With Extoolbarcrd1
	.ItemsDelimiter = ";"
	.Format = "1,-1,2"
	.Item(1).Caption = "Exit"
	.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

VB.NET for /COM

```
' Select event - Notifies once the user clicks the item.
Private Sub AxToolBarCRD1_Select(ByVal sender As System.Object, ByVal e As
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent) Handles
AxToolBarCRD1.Select
    With AxToolBarCRD1
        Debug.Print( "Select" )
        Debug.Print( e.selectedID )
    End With
End Sub

With AxToolBarCRD1
    .ItemsDelimiter = ";"
    .Format = "1,-1,2"
    .get_Item(1).Caption = "Exit"
    .get_Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
End With
```

C++

```
// Select event - Notifies once the user clicks the item.
void OnSelectToolBarCRD1(VARIANT   ID,VARIANT   SelectedID)
{
    /*
          Copy and paste the following directives to your header file as
          it defines the namespace 'EXTOOLBARCRDLib' for the library: 'ExToolBar.CRD
1.0 Control Library'
          #import <ExToolBar.CRD.dll>
          using namespace EXTOOLBARCRDLib;
     */
    EXTOOLBARCRDLib::IToolBarCRDPtr spToolBarCRD1 =
GetDlgItem(IDC_TOOLBARCRD1)->GetControlUnknown();
    OutputDebugStringW( L"Select" );
    OutputDebugStringW( L"SelectedID" );
}

EXTOOLBARCRDLib::IToolBarCRDPtr spToolBarCRD1 =
GetDlgItem(IDC_TOOLBARCRD1)->GetControlUnknown();
spToolBarCRD1->PutItemsDelimiter(L";");
spToolBarCRD1->PutFormat(L"1,-1,2");
spToolBarCRD1->GetItem(long(1))->PutCaption(L"Exit");
spToolBarCRD1->GetItem(long(2))->PutCaption(L"Item <b>A;Item <b>A#1;Item
<b>B#2;Item <b>C#3");
```

C++ Builder

```
// Select event - Notifies once the user clicks the item.
void __fastcall TForm1::ToolBarCRD1Select(TObject *Sender,Variant   ID,Variant
SelectedID)
{
    OutputDebugString( L"Select" );
    OutputDebugString( L"SelectedID" );
}

ToolBarCRD1->ItemsDelimiter = L";";
ToolBarCRD1->Format = L"1,-1,2";
ToolBarCRD1->Item[TVariant(1)]->Caption = L"Exit";
ToolBarCRD1->Item[TVariant(2)]->Caption = L"Item <b>A;Item <b>A#1;Item
<b>B#2;Item <b>C#3";
```

C#

```
// Select event - Notifies once the user clicks the item.
private void extoolbarcrd1_Select(object sender,object   ID,object   SelectedID)
{
    System.Diagnostics.Debug.Print( "Select" );
    System.Diagnostics.Debug.Print( SelectedID.ToString() );
}
//this.extoolbarcrd1.Select += new
exontrol.EXTOOLBARCRDLib.exg2antt.SelectEventHandler(this.extoolbarcrd1_Select);

extoolbarcrd1.ItemsDelimiter = ";";
extoolbarcrd1.Format = "1,-1,2";
```

```
extoolbarcrd1[1].Caption = "Exit";
extoolbarcrd1[2].Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3";
```

JScript/JavaScript

```
<BODY onload="Init()">
<SCRIPT FOR="ToolBarCRD1" EVENT="Select(ID,SelectedID)" LANGUAGE="JScript">
   alert( "Select" );
   alert( SelectedID );
</SCRIPT>

<OBJECT CLASSID="clsid:DD586AE6-F2A0-4308-8F34-8016B16F000E"
id="ToolBarCRD1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   ToolBarCRD1.ItemsDelimiter = ";";
   ToolBarCRD1.Format = "1,-1,2";
   ToolBarCRD1.Item(1).Caption = "Exit";
   ToolBarCRD1.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3";
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<SCRIPT LANGUAGE="VBScript">
Function ToolBarCRD1_Select(ID,SelectedID)
   With ToolBarCRD1
      alert( "Select" )
      alert( SelectedID )
   End With
End Function
```

```
</SCRIPT>

<OBJECT CLASSID="clsid:DD586AE6-F2A0-4308-8F34-8016B16F000E"
id="ToolBarCRD1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With ToolBarCRD1
        .ItemsDelimiter = ";"
        .Format = "1,-1,2"
        .Item(1).Caption = "Exit"
        .Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
// Select event - Notifies once the user clicks the item.
private void axToolBarCRD1_Select(object sender,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent e)
{
    System.Diagnostics.Debug.Print( "Select" );
    System.Diagnostics.Debug.Print( e.selectedID.ToString() );
}
//this.axToolBarCRD1.Select += new
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEventHandler(this.axToolBarCRD1_Sele


axToolBarCRD1.ItemsDelimiter = ";";
axToolBarCRD1.Format = "1,-1,2";
axToolBarCRD1[1].Caption = "Exit";
axToolBarCRD1[2].Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3";
```

X++ (Dynamics Ax 2009)

```
// Select event - Notifies once the user clicks the item.
void onEvent_Select(COMVariant   _ID,COMVariant   _SelectedID)
{
    ;
    print( "Select" );
    print( _SelectedID );
}

public void init()
{
    COM com_Item;
    anytype var_Item;
    ;

    super();

    extoolbarcrd1.ItemsDelimiter(";");
    extoolbarcrd1.Format("1,-1,2");
    var_Item =
COM::createFromObject(extoolbarcrd1.Item(COMVariant::createFromInt(1)));
com_Item = var_Item;
    com_Item.Caption("Exit");
    var_Item =
COM::createFromObject(extoolbarcrd1.Item(COMVariant::createFromInt(2)));
com_Item = var_Item;
    com_Item.Caption("Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3");
}
```

Delphi 8 (.NET only)

```
// Select event - Notifies once the user clicks the item.
procedure TWinForm1.AxToolBarCRD1_Select(sender: System.Object; e:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent);
begin
    with AxToolBarCRD1 do
    begin
        OutputDebugString( 'Select' );
        OutputDebugString( e.selectedID );
    end
end;

with AxToolBarCRD1 do
begin
    ItemsDelimiter := ';';
    Format := '1,-1,2';
    get_Item(TObject(1)).Caption := 'Exit';
    get_Item(TObject(2)).Caption := 'Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3';
end
```

Delphi (standard)

```
// Select event - Notifies once the user clicks the item.
procedure TForm1.ToolBarCRD1Select(ASender: TObject; ID : OleVariant;SelectedID :
OleVariant);
begin
    with ToolBarCRD1 do
    begin
        OutputDebugString( 'Select' );
        OutputDebugString( SelectedID );
    end
end;

with ToolBarCRD1 do
begin
    ItemsDelimiter := ';';
    Format := '1,-1,2';
    Item[OleVariant(1)].Caption := 'Exit';
    Item[OleVariant(2)].Caption := 'Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3';
end
```

VFP

```
*** Select event - Notifies once the user clicks the item. ***
```

```
LPARAMETERS ID,SelectedID
    with thisform.ToolBarCRD1
        DEBUGOUT( "Select" )
        DEBUGOUT( SelectedID )
    endwith

with thisform.ToolBarCRD1
    .ItemsDelimiter = ";"
    .Format = "1,-1,2"
    .Item(1).Caption = "Exit"
    .Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3"
endwith
```

dBASE Plus

```
/*
with (this.EXTOOLBAR_CRDACTIVEXCONTROL1.nativeObject)
    Select = class::nativeObject_Select
endwith
*/
// Notifies once the user clicks the item.
function nativeObject_Select(ID,SelectedID)
    oToolBarCRD = form.EXTOOLBAR_CRDACTIVEXCONTROL1.nativeObject
    ? "Select"
    ? Str(SelectedID)
return

local oToolBarCRD,var_Item,var_Item1

oToolBarCRD = form.EXTOOLBAR_CRDACTIVEXCONTROL1.nativeObject
oToolBarCRD.ItemsDelimiter = ";"
oToolBarCRD.Format = "1,-1,2"
// oToolBarCRD.Item(1).Caption = "Exit"
var_Item = oToolBarCRD.Item(1)
with (oToolBarCRD)
    TemplateDef = [dim var_Item]
    TemplateDef = var_Item
    Template = [var_Item.Caption = "Exit"]
endwith
// oToolBarCRD.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
var_Item1 = oToolBarCRD.Item(2)
with (oToolBarCRD)
    TemplateDef = [dim var_Item1]
    TemplateDef = var_Item1
    Template = [var_Item1.Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"]
endwith
```

XBasic (Alpha Five)

```
' Notifies once the user clicks the item.
function Select as v (ID  as  A,SelectedID  as  A)
    oToolBarCRD = topparent:CONTROL_ACTIVEX1.activex
    ? "Select"
    ? SelectedID
end function

Dim oToolBarCRD as P
Dim var_Item as local
Dim var_Item1 as local

oToolBarCRD = topparent:CONTROL_ACTIVEX1.activex
oToolBarCRD.ItemsDelimiter = ";"
oToolBarCRD.Format = "1,-1,2"
' oToolBarCRD.Item(1).Caption = "Exit"
var_Item = oToolBarCRD.Item(1)
oToolBarCRD.TemplateDef = "dim var_Item"
oToolBarCRD.TemplateDef = var_Item
oToolBarCRD.Template = "var_Item.Caption = `Exit`"

' oToolBarCRD.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
```

```
var_Item1 = oToolBarCRD.Item(2)
oToolBarCRD.TemplateDef = "dim var_Item1"
oToolBarCRD.TemplateDef = var_Item1
oToolBarCRD.Template = "var_Item1.Caption = `Item <b>A;Item <b>A#1;Item
<b>B#2;Item <b>C#3`"
```

Visual Objects

```
METHOD OCX_Exontrol1Select(ID,SelectedID) CLASS MainDialog
   // Select event - Notifies once the user clicks the item.
   OutputDebugString(String2Psz( "Select" ))
   OutputDebugString(String2Psz( AsString(SelectedID) ))
RETURN NIL


oDCOCX_Exontrol1:ItemsDelimiter := ";"
oDCOCX_Exontrol1:Format := "1,-1,2"
oDCOCX_Exontrol1:[Item,1]:Caption := "Exit"
oDCOCX_Exontrol1:[Item,2]:Caption := "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
```

PowerBuilder

```
/*begin event Select(any  ID,any  SelectedID) - Notifies once the user clicks the item.*/
/*
    oToolBarCRD = ole_1.Object
    MessageBox("Information",string( "Select" ))
    MessageBox("Information",string( String(SelectedID) ))
*/
/*end event Select*/

OleObject oToolBarCRD

oToolBarCRD = ole_1.Object
oToolBarCRD.ItemsDelimiter = ";"
oToolBarCRD.Format = "1,-1,2"
oToolBarCRD.Item(1).Caption = "Exit"
oToolBarCRD.Item(2).Caption = "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
```

Visual DataFlex

```
// Notifies once the user clicks the item.
Procedure OnComSelect Variant   llID Variant   llSelectedID
   Forward Send OnComSelect llID llSelectedID
   Showln "Select" llSelectedID
End_Procedure

Procedure OnCreate
   Forward Send OnCreate
   Set ComItemsDelimiter to ";"
   Set ComFormat to "1,-1,2"
   Variant voItem
   Get ComItem 1 to voItem
   Handle hoItem
   Get Create (RefClass(cComItem)) to hoItem
   Set pvComObject of hoItem to voItem
      Set ComCaption of hoItem to "Exit"
   Send Destroy to hoItem
   Variant voItem1
   Get ComItem 2 to voItem1
   Handle hoItem1
   Get Create (RefClass(cComItem)) to hoItem1
   Set pvComObject of hoItem1 to voItem1
      Set ComCaption of hoItem1 to "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"
   Send Destroy to hoItem1
End_Procedure
```

XBase++

```
PROCEDURE OnSelect(oToolBarCRD,ID,SelectedID)
```

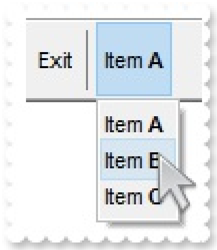
```
   DevOut( "Select" )
   DevOut( Transform(SelectedID,"") )
RETURN

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

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

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

   oToolBarCRD := XbpActiveXControl():new( oForm:drawingArea )
   oToolBarCRD:CLSID  := "Exontrol.ToolBar.CRD.1" /*{DD586AE6-F2A0-4308-8F34-
8016B16F000E}*/
   oToolBarCRD:create(,, {10,60},{610,370} )

      oToolBarCRD:Select := {|ID,SelectedID| OnSelect(oToolBarCRD,ID,SelectedID)}
/*Notifies once the user clicks the item.*/

      oToolBarCRD:ItemsDelimiter := ";"
      oToolBarCRD:Format := "1,-1,2"
      oToolBarCRD:Item(1):Caption := "Exit"
      oToolBarCRD:Item(2):Caption := "Item <b>A;Item <b>A#1;Item <b>B#2;Item
<b>C#3"

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

property ToolBarCRD.ItemValueDelimiter as String

Specifies the delimiter sequence for drop down value.

Type	Description
String	A String expression that specifies the delimiter for drop down values.

By default, the ItemsDelimiter property is "\r\n" (Chr(13) & Chr(10), Carriage return-linefeed combination). The ItemsDelimiter property specifies the delimiter sequence for drop down items. The ItemValueDelimiter property specifies the delimiter sequence for drop down value. The Caption property specifies the caption to be shown on the item. If the Caption property includes the ItemsDelimiter value, it indicates that the toolbar's item displays a drop down list, when user clicks the item. Each part of the Caption between two delimiters indicates an item in the drop down. For instance, "Item <b>A;Item <b>A#1;Item <b>B#2;Item <b>C#3" indicates that the toolbar's item displays the "Item **A**", and it's drop down list contains "Item **A**" with the value 1, "Item **B**" with the value 2 and "Item **C**" with the value 3.



property ToolBarCRD.Picture as IPictureDisp

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

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

By default, the control has no picture associated. Use the Picture property to assign your logo on the control's background. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. The BackColor property specifies a solid color to be shown on the control's background.

property ToolBarCRD.PictureDisplay as PictureDisplayEnum

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

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way the control arranges the picture on the control's background.

The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. By default, the control has no picture associated. Use the Picture property to assign your logo on the control's background. The BackColor property specifies a solid color to be shown on the control's background.

method ToolBarCRD.Refresh ()

Refreses the control.

Type	Description

Use the Refresh method to refresh the control. The BeginUpdate / EndUpdate method maintains performance when items are added to the control one at a time.

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

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

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

Use the ReplaceIcon property to add, remove or replace an icon in the control's images collection. Also, the ReplaceIcon property can clear the images collection. Use the Images method to attach a image list to the control. The user can add images at design time, by drag and drop files to control's images holder. The ShowImageList property available for the /COM shows or hides the control's images holder at design mode.

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

i = ExToolBarCRD1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added

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

i = ExToolBarCRD1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.

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

ExToolBarCRD1.ReplaceIcon 0, i, i specifies the index of icon removed.

The following VB clears the control's icons collection:

ExToolBarCRD1.ReplaceIcon 0, -1

property ToolBarCRD.ShowImageList as Boolean

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

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

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

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

Shows the specified tooltip at given position.

Type	Description
ToolTip as String	A String expression that indicates the description of the tooltip.
Title as Variant	If present, A String expression that indicates the title of the tooltip.
Alignment as Variant	A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing, the tooltip is aligned to the left/top corder.
X as Variant	A single that specifies the current X location of the mouse pointer. The x values is always expressed in screen coordinates. If missing or -1, the current mouse X position is used. A string expression that indicates the offset to move the tooltip window relative to the cursor position.
Y as Variant	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in screen coordinates. If missing or -1, the current mouse Y position is used. A string expression that indicates the offset to move the tooltip window relative to the cursor position.

Use the ShowToolTip method to display a custom tooltip. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

The Alignment parameter can be one of the followings:

- 0 - exTopLeft
- 1 - exTopRight
- 2 - exBottomLeft
- 3 - exBottomRight
- 0x10 - exCenter
- 0x11 - exCenterLeft
- 0x12 - exCenterRight
- 0x13 - exCenterTop

- 0x14 - exCenterBottom

*Use numeric values as strings for X and Y parameters, to move the tooltip window relative to the position of the cursor. For instance, ShowToolTp("text",,,***"11"***,***"12"***), means that the tooltip window is moved 11 pixels on the X axis, and 12 pixels on the Y axis, before showing it in the default position. In this case the X and Y parameters MUST be passed as strings not as LONG values*

property ToolBarCRD.Template as String

Specifies the control's template.

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

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

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

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

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

The Template 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 ToolBarCRD.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 ToolBarCRD.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 ToolBarCRD.ToolTipDelay as Long

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

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

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

property ToolBarCRD.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

Type	Description
IFontDisp	A Font object that defines the font to show the control's tooltip. You can use the <font> HTML tag to define a different font for parts of the tooltip.

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

property ToolBarCRD.ToolTipPopDelay as Long

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

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

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

property ToolBarCRD.ToolTipWidth as Long

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

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

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

property ToolBarCRD.Version as String

Retrieves the control's version.

Type	Description
String	A String expression that specifies the version of the control you are running.

The Version property specifies the version of the control you are running.

property ToolBarCRD.VisualAppearance as Appearance

Retrieves the control's appearance.

Type	Description
Appearance	The Appearance object holds a collection of skins.

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

ExToolBar.CRD 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: {DD586AE6-F2A0-4308-8F34-8016B16F000E}. The object's program identifier is: "Exontrol.ToolBar.CRD". The /COM object module is: "ExToolBar.CRD.dll"

The eXToolBar.CRD component is a graphical control element on which on-screen buttons, icons, menus, or other input or output elements are placed. The eXTolBar.CRD component lets the user changes its visual appearance using skins, each one providing an additional visual experience that enhances viewing pleasure. The eXToolBar.CRD component it's free to use so no nag screens, no limitation, no evaluation message.

The following table shows how you can create / access different type of objects (red items indicates the name of the property/method):

```
EXTOOLBARCRDLib.ToolBarCRD
   "Item(Variant)" -> EXTOOLBARCRDLib.Item
   "VisualAppearance" -> EXTOOLBARCRDLib.Appearance
```

The following table shows how you can create / access different type of objects (red items indicates the name of the property/method):

```
EXTOOLBARCRDLib.Appearance <- "VisualAppearance" of
EXTOOLBARCRDLib.ToolBarCRD
EXTOOLBARCRDLib.Item <- "Item(Variant)" of EXTOOLBARCRDLib.ToolBarCRD
```

The ExToolBar.CRD component supports the following events:

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

MouseIn	Occurs when the mouse enters the part.
MouseMove	Occurs when the user moves the mouse.
MouseOut	Occurs when the mouse exists the part.
MouseUp	Occurs when the user releases a mouse button.
Select	Notifies once the user clicks the item.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Type	Description
AnchorID as String	A string expression that indicates the identifier of the anchor
Options as String	A string expression that specifies options of the anchor element.

The control fires the AnchorClick event to notify that the user clicks an anchor element. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The AnchorClick event is fired only if prior clicking the control it shows the hand cursor. For instance, if the cell is disabled, the hand cursor is not shown when hovers the anchor element, and so the AnchorClick event is not fired. Use the FormatAnchor property to specify the visual effect for anchor elements. For instance, if the user clicks the anchor *<a1>anchor</a>*, the control fires the AnchorClick event, where the AnchorID parameter is 1, and the Options parameter is empty. Also, if the user clicks the anchor *<a 1;yourextradata>anchor</a>*, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "yourextradata".

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

C#
```
private void AnchorClick(object sender,string   AnchorID,string   Options)
{
}
```

VB
```
Private Sub AnchorClick(ByVal sender As System.Object,ByVal AnchorID As
String,ByVal Options As String) Handles AnchorClick
End Sub
```

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

C#
```
private void AnchorClick(object sender,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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: 
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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 
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,AnchorID,Options)

RETURN
```

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

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

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

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

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

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

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

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

event Click ()

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

Type	Description

The Click event is fired when the user releases the left mouse button over the control. The Select event notifies once the user clicks/select an item in the toolbar control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

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

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(oToolBarCRD)


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 user double clicks the control. Use the ItemFromPoint method to determine the cell over the cursor.

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,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,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.ToolBar.CRD.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.ToolBar.CRD.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event Event (EventID as Long)

Notifies the application once the control fires an event.

Type	Description
EventID as Long	A Long expression that specifies the identifier of the event. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties).

The Event notification occurs ANY time the control fires an event.

This is useful for X++ language, which does not support event with parameters passed by reference.

In X++ the "Error executing code: FormActiveXControl (data source), method ... called with invalid parameters" occurs when handling events that have parameters passed by reference. Passed by reference, means that in the event handler, you can change the value for that parameter, and so the control will takes the new value, and use it. The X++ is NOT able to handle properly events with parameters by reference, so we have the solution.

The solution is using and handling the Event notification and EventParam method., instead handling the event that gives the "invalid parameters" error executing code.

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,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,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,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,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,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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: AxEXTOOLBARCRDLib._IToolBarCRDEvents_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 AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,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. The Select event notifies once the user clicks/select an item in the toolbar control. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the ItemFromPoint property to get the item from point.

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, AxEXTOOLBARCRDLib._IToolBarCRDEvents_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:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,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.ToolBar.CRD.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.ToolBar.CRD.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event MouseIn (ID as Variant)

Occurs when the mouse enters the part.

Type	Description
ID as Variant	A Long expression that specifies the identifier of the item. The Item property accesses the Item object giving its identifier.

The MouseIn event occurs when the mouse enters the item. The MouseOut event occurs when the mouse exits the item. The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. Use the Background(exToolBarButtonHotBackColor) property to specify the visual appearance of the item when the cursor hovers it. Use the ItemFromPoint property to get the item from cursor.

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

C#
```
private void MouseIn(object sender,object   ID)
{
}
```

VB
```
Private Sub MouseIn(ByVal sender As System.Object,ByVal ID As Object) Handles
MouseIn
End Sub
```

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

C#
```
private void MouseIn(object sender,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_MouseInEvent e)
{
}
```

C++
```
void OnMouseIn(VARIANT   ID)
{
}
```

C++ Builder
```
void __fastcall MouseIn(TObject *Sender,Variant   ID)
{
}
```

Delphi
```
procedure MouseIn(ASender: TObject; ID : OleVariant);
begin
end;
```

Delphi 8 (.NET only)
```
procedure MouseIn(sender: System.Object; e:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_MouseInEvent);
begin
end;
```

Powe...
```
begin event MouseIn(any  ID)


end event MouseIn
```

VB.NET
```
Private Sub MouseIn(ByVal sender As System.Object, ByVal e As
AxEXTOOLBARCRDLib._IToolBarCRDEvents_MouseInEvent) Handles MouseIn
End Sub
```

VB6
```
Private Sub MouseIn(ByVal ID As Variant)
End Sub
```

VBA
```
Private Sub MouseIn(ByVal ID As Variant)
End Sub
```

VFP
```
LPARAMETERS ID
```

Xbas...
```
PROCEDURE OnMouseIn(oToolBarCRD,ID)

RETURN
```

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

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

VBSc...
```
<SCRIPT LANGUAGE="VBScript">
Function MouseIn(ID)
End Function
```

```
</SCRIPT>
```

Visual Data...

```
Procedure OnComMouseIn Variant   llID
    Forward Send OnComMouseIn llID
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseIn(ID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseIn(COMVariant   _ID)
{
}
```

XBasic

```
function MouseIn as v (ID  as  A)
end function
```

dBASE

```
function nativeObject_MouseIn(ID)
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. Gets which mouse button was pressed as 1 for Left Mouse Button, 2 for Right Mouse Button and 4 for Middle Mouse Button.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. The MouseIn event occurs when the mouse enters the item. The MouseOut event occurs when the mouse exits the item. Use the Background(exToolBarButtonHotBackColor) property to specify the visual appearance of the item when the cursor hovers it. Use the ItemFromPoint property to get the item from cursor.

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,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,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.ToolBar.CRD.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.ToolBar.CRD.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

```
return
```

event MouseOut (ID as Variant)

Occurs when the mouse exists the part.

Type	Description
ID as Variant	A Long expression that specifies the identifier of the item. The Item property accesses the Item object giving its identifier.

The MouseOut event occurs when the mouse exits the item. The MouseIn event occurs when the mouse enters the item. The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. Use the Background(exToolBarButtonHotBackColor) property to specify the visual appearance of the item when the cursor hovers it. Use the ItemFromPoint property to get the item from cursor.

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

C#
```
private void MouseOut(object sender,object   ID)
{
}
```

VB
```
Private Sub MouseOut(ByVal sender As System.Object,ByVal ID As Object) Handles
MouseOut
End Sub
```

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

C#
```
private void MouseOut(object sender,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_MouseOutEvent e)
{
}
```

C++
```
void OnMouseOut(VARIANT   ID)
{
}
```

C++ Builder
```
void __fastcall MouseOut(TObject *Sender,Variant   ID)
{
}
```

Delphi

```
procedure MouseOut(ASender: TObject; ID : OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseOut(sender: System.Object; e:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_MouseOutEvent);
begin
end;
```

Powe…

```
begin event MouseOut(any  ID)

end event MouseOut
```

VB.NET

```
Private Sub MouseOut(ByVal sender As System.Object, ByVal e As
AxEXTOOLBARCRDLib._IToolBarCRDEvents_MouseOutEvent) Handles MouseOut
End Sub
```

VB6

```
Private Sub MouseOut(ByVal ID As Variant)
End Sub
```

VBA

```
Private Sub MouseOut(ByVal ID As Variant)
End Sub
```

VFP

```
LPARAMETERS ID
```

Xbas…

```
PROCEDURE OnMouseOut(oToolBarCRD,ID)

RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseOut(ID)
End Function
```

```
</SCRIPT>
```

Visual Data...

```
Procedure OnComMouseOut Variant   llID
    Forward Send OnComMouseOut llID
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseOut(ID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseOut(COMVariant   _ID)
{
}
```

XBasic

```
function MouseOut as v (ID  as  A)
end function
```

dBASE

```
function nativeObject_MouseOut(ID)
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. The Select event notifies once the user clicks/select an item in the toolbar control. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the ItemFromPoint property to get the item from point.

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, AxEXTOOLBARCRDLib._IToolBarCRDEvents_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:
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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
AxEXTOOLBARCRDLib._IToolBarCRDEvents_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(oToolBarCRD,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.ToolBar.CRD.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.ToolBar.CRD.1::OLE_YPOS_PIXELS)
end function
```

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

event Select (ID as Variant, SelectedID as Variant)

Notifies once the user clicks the item.

Type	Description
ID as Variant	A Long expression that specifies the identifier of the item being clicked/selected. The Item property accesses the Item object giving its identifier.
SelectedID as Variant	A Long expression that specifies the value of the drop down item being clicked/selected.

The Select event notifies once the user clicks/select an item in the toolbar control. The Click event is fired when the user releases the left mouse button over the control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

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

C#
```
private void Select(object sender,object   ID,object   SelectedID)
{
}
```

VB
```
Private Sub Select(ByVal sender As System.Object,ByVal ID As Object,ByVal
SelectedID As Object) Handles Select
End Sub
```

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

C#
```
private void Select(object sender,
AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent e)
{
}
```

C++
```
void OnSelect(VARIANT   ID,VARIANT   SelectedID)
{
}
```

C++ Builder

```
void __fastcall Select(TObject *Sender,Variant   ID,Variant   SelectedID)
{
}
```

Delphi

```
procedure Select(ASender: TObject; ID : OleVariant;SelectedID : OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Select(sender: System.Object; e: AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent);
begin
end;
```

Powe…

```
begin event Select(any  ID,any  SelectedID)

end event Select
```

VB.NET

```
Private Sub Select(ByVal sender As System.Object, ByVal e As AxEXTOOLBARCRDLib._IToolBarCRDEvents_SelectEvent) Handles Select
End Sub
```

VB6

```
Private Sub Select(ByVal ID As Variant,ByVal SelectedID As Variant)
End Sub
```

VBA

```
Private Sub Select(ByVal ID As Variant,ByVal SelectedID As Variant)
End Sub
```

VFP

```
LPARAMETERS ID,SelectedID
```

Xbas…

```
PROCEDURE OnSelect(oToolBarCRD,ID,SelectedID)

RETURN
```

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

Java…

```
<SCRIPT EVENT="Select(ID,SelectedID)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Select(ID,SelectedID)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComSelect Variant   llID Variant   llSelectedID
    Forward Send OnComSelect llID llSelectedID
End_Procedure
```

Visual Objects

```
METHOD OCX_Select(ID,SelectedID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Select(COMVariant   _ID,COMVariant   _SelectedID)
{
}
```

XBasic

```
function Select as v (ID  as  A,SelectedID  as  A)
end function
```

dBASE

```
function nativeObject_Select(ID,SelectedID)
return
```