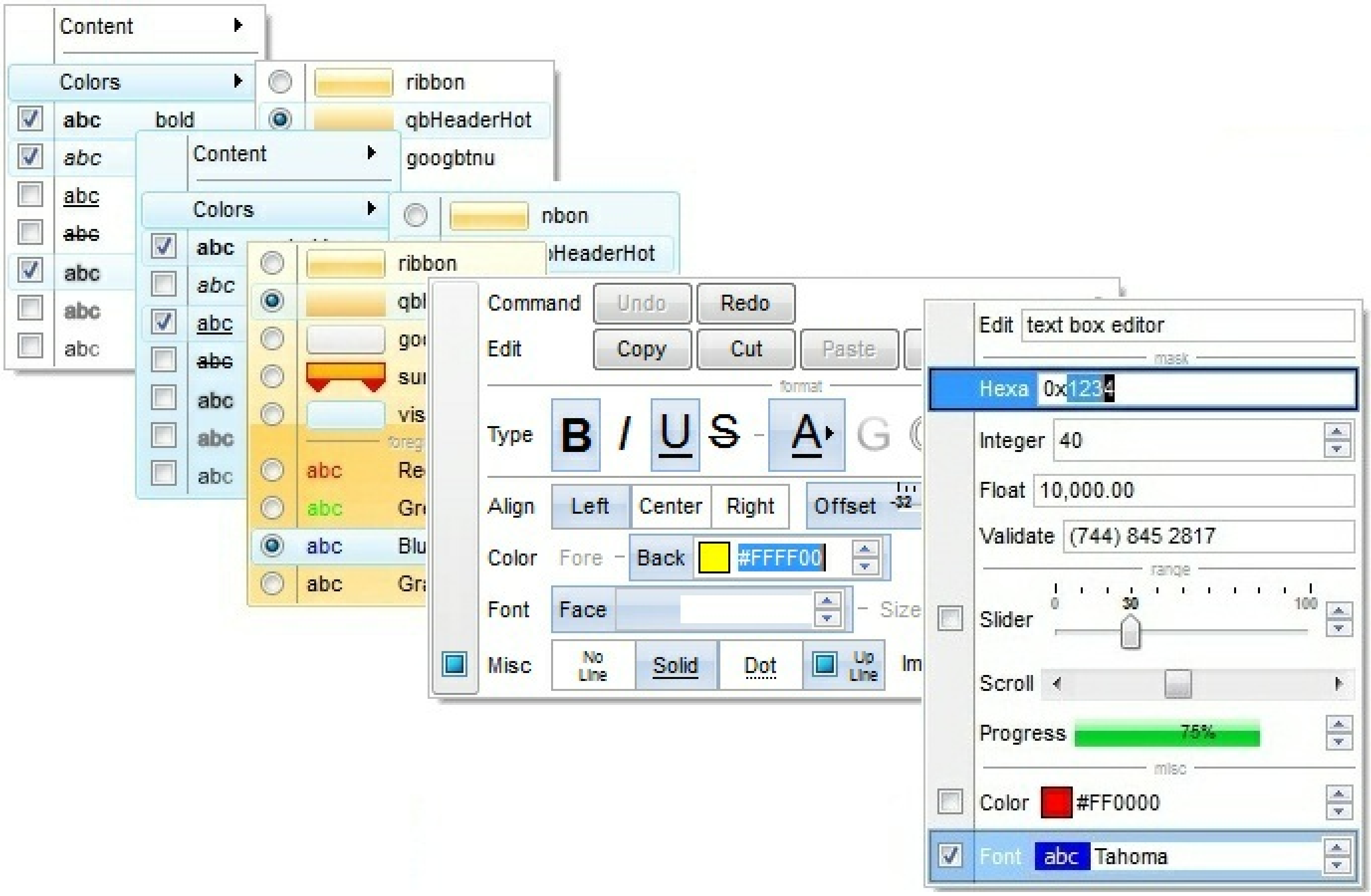
ExContextMenu

The eXContextMenu component displays and handles a context menu (also called contextual, shortcut, and popup or pop-up menu). A context menu is a menu in a graphical user interface (GUI) that appears upon user interaction, such as a right-click mouse operation. The eXContextMenu component is written from scratch, and does NOT use the system's popup menu. For instance, the /NET's System.Windows.Controls.ContextMenu does not support a modal form, so you have to assign a handler for each item, instead the eXContextMenu component waits for the user to make the selection, and returns the selected values. Also another major difference is that the System.Windows.Controls.ContextMenu is closed once any item is clicked, while in the eXContextMenu component, this is not required, so you can check multiple check boxes, and when you click outside, the Select method returns the selected values.

The features include:

- Skinnable Interface support (ability to apply a skin to any item)
- Keyboard and Mouse Wheel support
- Check box / Radio button support
- Ability to use any ActiveX control inside sub menus
- Ability to assign EDIT, MASK, COLOR, FONT, SPIN, SLIDER, SCROLLBAR, PROGRESS, ... fields to any item
- Multi-lines HTML Tooltip support for any item
- Ability to wait for user to select one or more values, as modal
- Ability to specify when to close the context menu, not necessary a single click
- Incremental Search / Shortcut Keys support (Ability to display/filter the items that match the typing characters)
- Ability to load/save the menu from strings like "Item 1[bld],Item2[chk]", without having to call the Add method
- Ability to group the items, so a sub-menu can be shown in the current item
- Ability to define the round frame for the context menu, using the EBN objects.
- Images / Icons support
- Partially Translucent support
- Ability to scroll the menu items
- HTML support, including text decorations like shadow, outline or gradient text
- Ability to query at once the entire menu for all items being checked, or radio buttons, that contains EDIT fields, and so on



Edit text box editor

Hexa 0x1234

Integer 40

Float 10,000.00

Validate (744) 845 2817

Color Back #FFFF00

Color #FF0000

Font abc Tahoma

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

How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

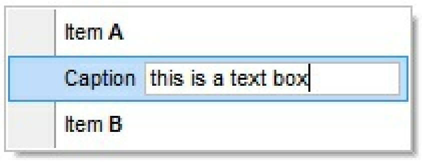
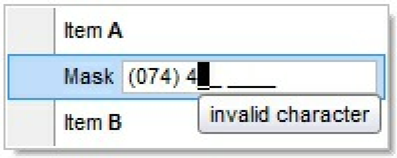
Specifies the object's Alignment. The Alignment property specifies the item's alignment. The Caption property supports built-in HTML format, so you can use the <c> to centers the item's caption or <r> to align to the right the item's caption. The AlignmentEnum expression supports the following values:

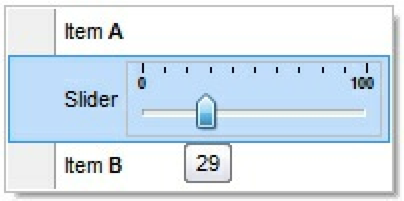
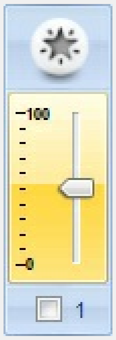
Name	Value	Description
exLeft	0	Left
exCenter	1	Center
exRight	2	Right

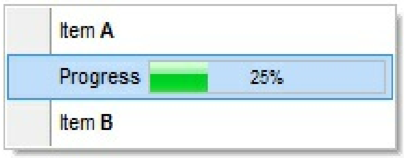
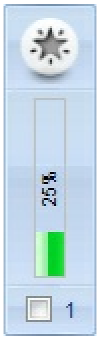
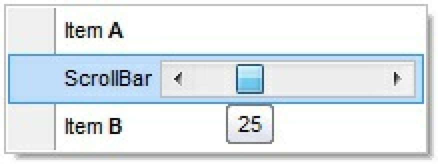
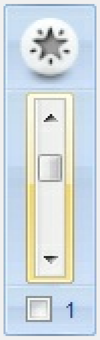
constants AllowEditEnum

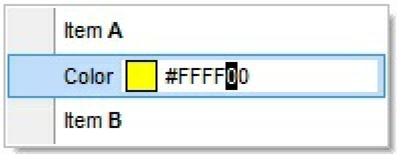
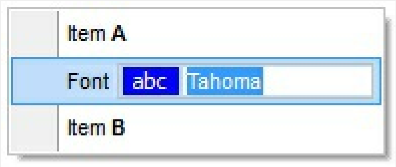
The AllowEditEnum type specifies the type of editors that can be associated with the item. The AllowEdit property associates an editor to the current item. The EditCaption property specifies the value to show in the edit field. The EditWidth property specifies the size/width of the edit field inside the item. The EditBorder property specifies whether the edit shows a border around it. The EditOption property specifies different options to be used for a specified edit field. The control fires the EditChange event when the user changes the edit's caption. Use the ShowLocalPopup property to provide a drop down list. The ShowAsButton property specifies the whether the current item displays a button or a select button (drop down).

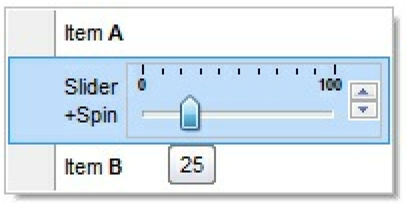
Currently, the supported editors are:

Name	Value	Description
exItemDisableEdit	0	No editor is assigned to the current item.
exItemEditText	1	A text-box editor is assigned to the current item. The exItemEditText can be combined with the exItemEditReadOnly or exItemEditSpin flags. 
exItemEditMask	2	A masked text-box editor is assigned to the current item. The EditMask property specifies the mask of the edit field. The EditValue property specifies the value of the edit field, without the masking characters. The EditOption(exEditMaskFloat) specifies whether the edit field mask a floating/decimal/integer point number. The exItemEditMask can be combined with the exItemEditReadOnly or exItemEditSpin flags.  Item A Mask (074) 4█_ ____ invalid character Item B
		A slider editor is assigned to the current item. The exItemEditSlider can be combined with the exItemEditReadOnly, exItemEditVertical or exItemEditSpin flags. The EditValue property

exItemEditSlider	3	indicates the current slider position/value. • The EditOption(exEditMinValue) / EditOption(exEditMaxValue) specifies the limits values of the slider editor. • The EditOption(exEditTickStyle) property specifies the way the ticks are shown on the slider. • The EditOption(exEditTickFrequency) property specifies the frequency to show the ticks on the slider control. • The EditOption(exEditTickLabel) property specifies labels to be shown on the slider's ticks. • The EditOption(exEditSmallChange) property specifies the amount by which the edit's position changes when the user presses an arrow key. • The EditOption(exEditLargeChange) property specifies the amount by which the edit's position changes when the user presses an CTRL + arrow key. • The EditOption(exEditChangeToolTip) property specifies the specifies the expression that determines the HTML tooltip to be shown when the item's value is changed.  If exItemEditSlider flag is combined with the exItemEditVertical you can get: 

exItemEditProgress	4	A progress editor is assigned to the current item. The exItemEditProgress can be combined with the exItemEditReadOnly, exItemEditVertical or exItemEditSpin flags. The EditValue property indicates the current progress position/value.  If exItemEditProgress flag is combined with the exItemEditVertical you can get: 
exItemEditScrollBar	5	A scrollbar editor is assigned to the current item. The exItemEditScrollBar can be combined with the exItemEditReadOnly, exItemEditVertical or exItemEditSpin flags. The EditValue property indicates the current scroll position/value.  If exItemEditScrollBar flag is combined with the exItemEditVertical you can get:  • The EditOption(exEditMinValue) /

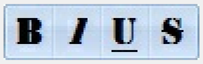
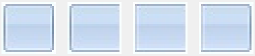
		EditOption(exEditMaxValue) specifies the limits values of the scroll editor. - The EditOption(exEditSmallChange) property specifies the amount by which the edit's position changes when the user presses an arrow key. - The EditOption(exEditLargeChange) property specifies the amount by which the edit's position changes when the user presses an CTRL + arrow key. - The EditOption(exEditChangeToolTip) property specifies the specifies the expression that determines the HTML tooltip to be shown when the item's value is changed.
exItemEditColor	6	A color editor is assigned to the current item. The exItemEditColor can be combined with the exItemEditReadOnly or exItemEditSpin flags. The EditValue property indicates the current color value.  Item A Color #FFFF00 Item B
exItemEditFont	7	A font editor is assigned to the current item. The exItemEditFont can be combined with the exItemEditReadOnly or exItemEditSpin flags. The EditCaption property indicates the current font name.  Item A Font abc Tahoma Item B
exItemEditReadOnly	256	Disables the current's item editor. This flag can be combined with any other option.
		A spin editor is assigned to the current item. This flag can be combined with any other option. The following picture combines a exItemEditSlider with the exItemEditSpin

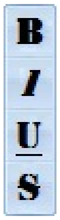
exItemEditSpin	512	 • The EditOption(exEditSpinStep) specifies the step to advance when user clicks the editor's spin.
exItemEditVertical	1024	The editor is vertically oriented. You can combine this flag with exItemEditSlider, exItemEditProgress and exItemEditScrollBar

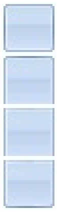
constants BackgroundPartEnum

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

Name	Value	Description
exToolTipAppearance	64	Specifies the visual appearance of the borders of the tooltips.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.
exCheckBoxState0	70	Specifies the visual appearance for the check box in 0 state (unchecked).
exCheckBoxState1	71	Specifies the visual appearance for the check box in 1 state (checked).
exCheckBoxState2	72	Specifies the visual appearance for the check box in 2 state (partial, not used).
exRadioButtonState0	73	Specifies the visual appearance for the radio button in 0 state (unchecked).
exRadioButtonState1	74	Specifies the visual appearance for the radio button in 1 state (checked).
exMenuFlatLineColor	100	Specifies the color to show the vertical line on flat appearance.
exMenuScrollBackColor	101	Specifies the background color to show the menu's scroll bars.
exMenuSelBorderColor	102	Specifies the color to show the frame arround the selected item.
exMenuSeparatorItem	103	Specifies the color to show the separator item.
exMenuButtonItem	104	Specifies the visual appearance for an item, when the Appearance property is Button.
		Specifies the visual appearance/solid color of the frame around the grouping items, when its group includes a single item. The GroupPopup property specifies the way the item's submenu are grouped.

exGroupPopupFrameSingle	105	Use the exGroupPopupFrameSingle, exGroupPopupFrameHStart, exGroupPopupFrameHIntermediate and exGroupPopupFrameHEnd to specify a different visual appearance for the frame around grouping items (horizontally). The following screen shot shows the grouping items with an EBN frame:  which has been created using the following 4 EBNs: 
exGroupPopupFrameHStart	106	Specifies the visual appearance/solid color of the frame around the first item (horizontally arranged), when the its group includes more items. The GroupPopup property specifies the way the item's submenu are grouped. Use the exGroupPopupFrameSingle, exGroupPopupFrameHStart, exGroupPopupFrameHIntermediate and exGroupPopupFrameHEnd to specify a different visual appearance for the frame around grouping items (horizontally).
exGroupPopupFrameHIntermediate	107	Specifies the visual appearance/solid color of the frame around an intermediate item (not start or end item, horizontally arranged), when the its group includes more items. The GroupPopup property specifies the way the item's submenu are grouped. Use the exGroupPopupFrameSingle, exGroupPopupFrameHStart, exGroupPopupFrameHIntermediate and exGroupPopupFrameHEnd to specify a different visual appearance for the frame around grouping items (horizontally).
		Specifies the visual appearance/solid color of the frame around the last item (not start or intermediate item, horizontally arranged), when the its group includes more items. The GroupPopup

exGroupPopupFrameHEnd	108	property specifies the way the item's submenu are grouped. Use the exGroupPopupFrameSingle, exGroupPopupFrameHStart, exGroupPopupFrameHIntermediate and exGroupPopupFrameHEnd to specify a different visual appearance for the frame around grouping items (horizontally).
exGroupPopupFrameSolid	109	Specifies the solid color of the frame around the grouping items.
exMenuHotBackColor	110	Specifies the visual appearance/color to show the item from the cursor.
exMenuHotForeColor	111	Specifies the foreground color to show the item from the cursor.
exMenuSelHotBackColor	112	Specifies the visual appearance/color to show the selected item from the cursor.
exMenuSelHotForeColor	113	Specifies the foreground color to show the selected item from the cursor.
exMenuSeparatorSelectButton	114	Specifies the visual appearance/color to show the separator between select and drop down button.
exMenuSeparatorSelectButtonBottom	115	Specifies the visual appearance/color to show the separator between select and drop down button (show the drop-down button to the bottom).
exGroupPopupFrameVStart	116	Specifies the visual appearance/solid color of the frame around the first item (vertically arranged), when the its group includes more items. The GroupPopup property specifies the way the item's submenu are grouped. Use the exGroupPopupFrameVStart, exGroupPopupFrameVIntermediate and exGroupPopupFrameVEnd to specify a different visual appearance for the frame around grouping items (vertically). The following screen shot shows the grouping items with an EBN frame: 

		which has been created using the following 4 EBNs: 
exGroupPopupFrameVIntermediate	117	Specifies the visual appearance/solid color of the frame around an intermediate item (not start or end item, vertically arranged), when the its group includes more items. The GroupPopup property specifies the way the item's submenu are grouped. Use the exGroupPopupFrameVStart, exGroupPopupFrameVIntermediate and exGroupPopupFrameVEnd to specify a different visual appearance for the frame around grouping items (vertically).
exGroupPopupFrameVEnd	118	Specifies the visual appearance/solid color of the frame around the last item (not intermediate or end item, vertically arranged), when the its group includes more items. The GroupPopup property specifies the way the item's submenu are grouped. Use the exGroupPopupFrameVStart, exGroupPopupFrameVIntermediate and exGroupPopupFrameVEnd to specify a different visual appearance for the frame around grouping items (vertically).

constants CloseOnClickEnum

The CloseOnClickEnum type specifies when the user can close the context menu. The CloseOnClick property specifies how the context menu is closed when user clicks an item. The CloseOnClickEnum type supports the following values:

Name	Value	Description
exCloseOnClick	0	The popup menu is closing when the user clicks an item.
exCloseOnDblClick	1	The popup menu is closing when the user double clicks an item.
exCloseOnClickOutside	2	The popup menu is closing when the user clicks outside of the menu.
exCloseOnNonClickable	3	The popup menu is closing when the user clicks a non-clickable item (regular items). Here's the list of clickable items: • separator items • item that hosts a sub-menu (popup item) • disabled item • check or radio items For instance, clicking a check-box item will makes the check box to change its state instead closing the context menu.

constants CloseOnEnum

The CloseOnEnum type specifies how an item that contains an ActiveX inside is close. The CloseOn property indicates how the user closes the context menu when an inside ActiveX control is clicked. The CloseOnEnum type supports the following values:

Name	Value	Description
exUser	0	The user is responsible for closing the popup menu.
exLButtonDown	513	The popup menu is closed when user presses the left mouse button over the ActiveX control.
exLButtonUp	514	The popup menu is closed when user releases the left mouse button over the ActiveX control.
exLButtonDblClk	515	The popup menu is closed when user double click the ActiveX control.
exRButtonDown	516	The popup menu is closed when user right clicks the ActiveX control.
exRButtonUp	517	The popup menu is closed when user releases the right mouse button over the ActiveX control.
exRButtonDblClk	518	The popup menu is closed when user double click the right mouse button in the ActiveX control.
exMButtonDown	519	The popup menu is closed when user clicks the middle mouse button in the ActiveX control.
exMButtonUp	520	The popup menu is closed when user releases the middle mouse button in the ActiveX control.
exMButtonDblClk	521	The popup menu is closed when user double clicks the middle mouse button in the ActiveX control.
exClick	61441	The popup menu is closed when user presses any of the mouse buttons in the ActiveX control.
exDblClick	61442	The popup menu is closed when user double clicks any of the mouse buttons in the ActiveX control.

constants EditBorderEnum

Specifies the type of the border around the edit control inside the item. Use the AllowEdit property to assign a single edit control to an item. Use the EditBorder property to specify the type of the border for the edit control inside the item.

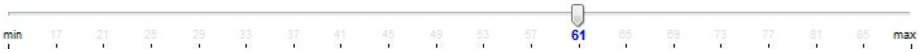
Name	Value	Description
exEditBorderNone	0	No border.
exEditBorderInset	-1	Inset border.
exEditBorderSingle	1	Single border.

constants EditOptionEnum

The EditOptionEnum type specifies different options to be set / get for giving editor. The EditOption property specifies different options to be used for a specified edit field. The AllowEdit property associates an editor to the current item. The EditCaption property specifies the value to show in the edit field. The EditWidth property specifies the size/width of the edit field inside the item. The EditBorder property specifies whether the edit shows a border around it. The control fires the EditChange event when the user changes the edit's caption.

The control supports the following options:

Name	Value	Description
exEditMinValue	1	Specifies the minimum value for the item's edit field. By default, the exEditMinValue option is 0. This option is valid for editors like: exItemEditSlider, exItemEditScrollBar (long expression)
exEditMaxValue	2	Specifies the maximum value for the item's edit field. By default, the exEditMinValue option is 100. This option is valid for editors like: exItemEditSlider, exItemEditScrollBar (long expression)
exEditTickStyle	3	Specifies where the ticks appears on the edit field. By default, the exEditTickStyle option is 1. The value of this option could be one of the following: • **0** (exBottomRight), The ticks are displayed on the bottom/right side • **1** (exTopLeft), The ticks are displayed on the top/left side • **2** (exBoth), The ticks are displayed on the both side • **3** (exNoTicks), No ticks are displayed This option is valid for editors like: exItemEditSlider (long expression)

exEditTickFrequency	4	Indicates the ratio of ticks on the control. By default, the exEditTickFrequency option is 10. This option is valid for editors like: exItemEditSlider (long expression)
		Specifies the expression that determines the HTML labels to be shown on ticks.  min 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 max For instance: - "value", shows the values for each tick. - " (value=current ? '<font ;12><fgcolor=FF0000>' : '') + value", shows the current slider's position with a different color and font. - "value = current ? value : ''", shows the value for the current tick only. - "(value = current ? '<b><font ;10>' : '') + (value array 'ab bc cd de ef fg gh hi ij jk kl' split ' ')" displays different captions for slider's values. The option supports the following keywords: - **value** gets the slider's position to be displayed - **current** gets the current slider's value. - **vmin** gets the slider's minimum value. - **vmax** gets the slider's maximum value. - **smin** gets the slider's selection minimum value. - **smax** gets the slider's selection maximum value. *The supported binary arithmetic operators are:* - ***** (multiplicity operator), priority 5 - **/** (divide operator), priority 5 - **mod** (reminder operator), priority 5 - **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type) - **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

"expression array (c1,c2,c3,...cn)"

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')"* is equivalent with *"month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'*

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

"expression in (c1,c2,c3,...cn)"

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case* *(default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
    - #4/1/2009#, from hours 06:00 AM to 12:00 PM
    - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
    - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

    - 0 - empty (not initialized)
    - 1 - null
    - 2 - short
    - 3 - long
    - 4 - float

		○ 5 - double ○ 6 - currency ○ 7 - date ○ 8 - string ○ 9 - object ○ 10 - error ○ 11 - boolean ○ 12 - variant ○ 13 - any ○ 14 - decimal ○ 16 - char ○ 17 - byte ○ 18 - unsigned short ○ 19 - unsigned long ○ 20 - long on 64 bits ○ 21 - unsigned long on 64 bites • **str** (unary operator) converts the expression to a string • **dbl** (unary operator) converts the expression to a number • **date** (unary operator) converts the expression to a date, based on your regional settings • **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS.
exEditTickLabel	5	*Other known operators for numbers are:* • **int** (unary operator) retrieves the integer part of the number • **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2 • **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument • **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2 • value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English

format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep* with the following meanings:

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and

Language Options" is using.
    - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
        - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
        - 1 - Negative sign, number; for example, -1.1
        - 2 - Negative sign, space, number; for example, - 1.1
        - 3 - Number, negative sign; for example, 1.1-
        - 4 - Number, space, negative sign; for example, 1.1 -
    - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a

string ends with specified string

- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.
- a **split** b, splits the a using the separator b, and returns an array. For instance, the "weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' **split ' '**" gets the weekday as string. This operator can be used with the array

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)

- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

The The <VALUE> of [ticklabel] option can display labels using the following built-in HTML tags:

- **<b></b>** displays the text in **bold**.
- **<i></i>** displays the text in *italics*.
- **<u></u>** underlines the text.
- **<s></s>** ~~Strike~~-through text
- **<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>** displays text with a specified foreground color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<bgcolor=RRGGBB></bgcolor>** displays text with a specified background color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<br>** a forced line-break
- **<solidline>** The next line shows a solid-line on

top/bottom side. If has no effect for a single line caption.

- **<dotline>** The next line shows a dot-line on top/bottom side. If has no effect for a single line caption.
- **<upline>** The next line shows a solid/dot-line on top side. If has no effect for a single line caption.
- **<r>** Right aligns the text
- **<c>** Centers the text
- **<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, in UNICODE configuration. 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; (string expression)
exEditSmallChange	6	The amount by which the edit's position changes when the user presses an arrow key. By default, the exEditSmallChange option is 1. This option is valid for editors like: exItemEditSlider, exItemEditScrollBar (long expression)
exEditLargeChange	7	The amount by which the edit's position changes when the user presses an CTRL + arrow key. By default, the exEditLargeChange option is 5. This option is valid for editors like: exItemEditSlider, exItemEditScrollBar (long expression)
		Specifies the expression that determines the HTML tooltip to be shown when the item's value is changed. By default, the exEditChangeToolTip option is "value". This option is valid for editors like: exItemEditSlider, exItemEditScrollBar The option supports the following keywords: • **value** gets the slider's position to be displayed *The supported binary arithmetic operators are:* • * (multiplicity operator), priority 5 • **/** (divide operator), priority 5 • **mod** (reminder operator), priority 5 • **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type) • **-** (subtraction operator), priority 4 *The supported unary boolean operators are:* • **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

"expression array (c1,c2,c3,...cn)"

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')"* is equivalent with *"month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'*

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

"expression in (c1,c2,c3,...cn)"

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *" (expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
    - #4/1/2009#, from hours 06:00 AM to 12:00 PM
    - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
    - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

    - 0 - empty (not initialized)
    - 1 - null
    - 2 - short
    - 3 - long
    - 4 - float
    - 5 - double
    - 6 - currency
    - 7 - date
    - 8 - string

exEditChangeToolTip	8	○ 9 - object ○ 10 - error ○ 11 - boolean ○ 12 - variant ○ 13 - any ○ 14 - decimal ○ 16 - char ○ 17 - byte ○ 18 - unsigned short ○ 19 - unsigned long ○ 20 - long on 64 bits ○ 21 - unsigned long on 64 bites • **str** (unary operator) converts the expression to a string • **dbl** (unary operator) converts the expression to a number • **date** (unary operator) converts the expression to a date, based on your regional settings • **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. *Other known operators for numbers are:* • **int** (unary operator) retrieves the integer part of the number • **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2 • **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument • **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2 • value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for

some invalid parameter, the value is displayed with no formatting.

The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep* with the following meanings:

- *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from

    "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
        - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
        - 1 - Negative sign, number; for example, -1.1
        - 2 - Negative sign, space, number; for example, - 1.1
        - 3 - Number, negative sign; for example, 1.1-
        - 4 - Number, space, negative sign; for example, 1.1 -
    - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of

the string

- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.
- a **split** b, splits the a using the separator b, and returns an array. For instance, the "weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' **split** ' '" gets the weekday as string. This operator can be used with the array

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)

- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

The The <VALUE> of [ticklabel] option can display labels using the following built-in HTML tags:

- **<b></b>** displays the text in **bold**.
- **<i></i>** displays the text in *italics*.
- **<u></u>** underlines the text.
- **<s></s>** ~~Strike~~-through text
- **<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>** displays text with a specified foreground color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<bgcolor=RRGGBB></bgcolor>** displays text with a specified background color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<br>** a forced line-break
- **<solidline>** The next line shows a solid-line on top/bottom side. If has no effect for a single line caption.
- **<dotline>** The next line shows a dot-line on top/bottom side. If has no effect for a single

line caption.

- **<upline>** The next line shows a solid/dot-line on top side. If has no effect for a single line caption.
- **<r>** Right aligns the text
- **<c>** Centers the text
- **<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, in UNICODE configuration. 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;

(string expression)

exEditMaskFloat	9	Specifies whether the mask field masks a floating point number. By default, the exEditMaskFloat is False. This flag is valid only for editors of exItemEditMask type. (boolean expression)
exEditSpinStep	10	Specifies the step to advance when user clicks the editor's spin. By default, the exEditSpinStep is 1. This flag is valid only for editors of exItemEditSpin type. (long expression)

constants GroupPopupEnum

The GroupPopupEnum type specifies whether the sub-menu of the current item is shown as grouped. The GroupPopup property specifies whether the sub-menu of the current item is shown as grouped. The GroupPopupEnum type supports the following values:

Name	Value	Description
exNoGroupPopup	0	No grouping is performed on the popup item.
exGroupPopup	1	Groups and displays the sub-menu items on the current item, arranged from left to right (by default, the items are horizontally arranged).
exNoGroupPopupFrame	2	Prevents showing the frame around each grouping item. If the exNoGroupPopupFrame flag is not present, the control shows a frame around the grouping items. The frame around grouping items can be a solid box or EBN frames. The Background(exGroupPopupFrameSolid) specifies the color to show the frames around grouping items, while the The Background(exGroupPopupFrameSingle), Background(exGroupPopupFrameHStart), Background(exGroupPopupFrameHIntermediate), Background(exGroupPopupFrameHEnd) specifies the EBN to be shown for single element, starting element/item of the group, intermediate elements, and ending element/item. The exGroupPopupFrameHStart, exGroupPopupFrameHIntermediate and exGroupPopupFrameHEnd are valid for horizontally grouping items (exGroupPopupVertical flag is not present).
exGroupPopupCenter	4	Shows the grouping popup aligned to the center of the current item.
exGroupPopupRight	8	Shows the grouping popup aligned to the right of the current item.
exGroupPopupEqualWidth	16	Shows the items that make the group of the same width.
exGroupPopupEqualHeight	32	Shows the items that make the group of the same height.
		Forces the grouping items to show the solid frame

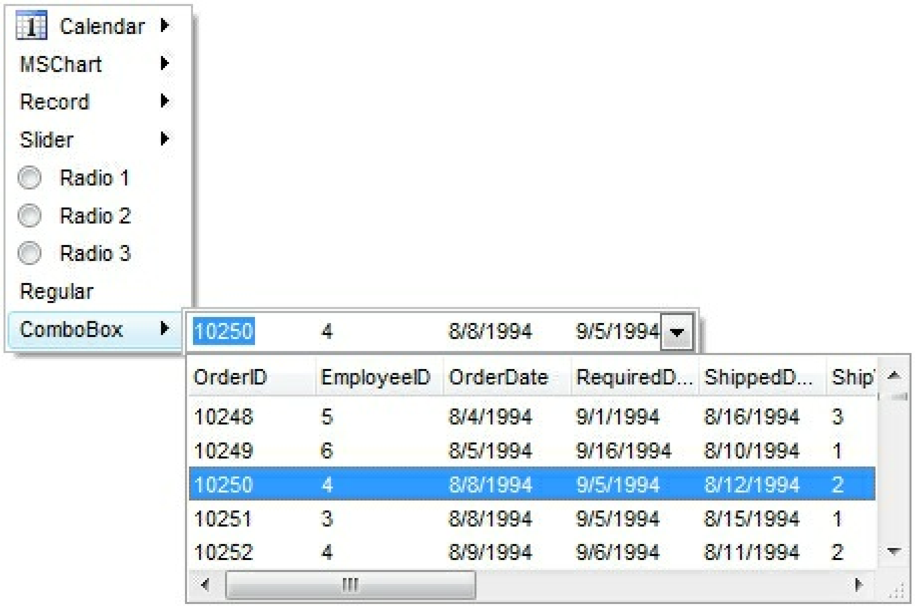
exGroupPopupFrameSolidBox	64	(exGroupPopupFrameSolid) rather than EBN frame. The exGroupPopupFrameSolidBox flag can be combined with the exGroupPopupFrameThickBox to specify a ticker solid frame. This flag has no effect if the exNoGroupPopupFrame is present.
exGroupPopupFrameThickBox	128	Specifies that the grouping items shows a ticker frame. The exGroupPopupFrameThickBox flag can be combined with the exGroupPopupFrameSolidBox to specify a ticker solid frame. This flag has no effect if the exNoGroupPopupFrame is present.
exGroupPopupVertical	256	Arranges vertically the items that compose the group (by default, the items are horizontally arranged). If the exNoGroupPopupFrame flag is not present, the control shows a frame around the grouping items. The frame around grouping items can be a solid box or EBN frames. The Background(exGroupPopupFrameSolid) specifies the color to show the frames around grouping items, while the The Background(exGroupPopupFrameSingle), Background(exGroupPopupFrameVStart), Background(exGroupPopupFrameVIntermediate), Background(exGroupPopupFrameVEnd) specifies the EBN to be shown for single element, starting element/item of the group, intermediate elements, and ending element/item. The exGroupPopupFrameVStart, exGroupPopupFrameVIntermediate and exGroupPopupFrameVEnd are valid for vertically grouping items (exGroupPopupVertical flag is present).

constants MenuAppearanceEnum

The MenuAppearanceEnum type indicates the menu's appearance. The MenuAppearanceEnum type supports the following values:

<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>exMenuNormal</td><td>0</td><td>

The BackColor property specifies the menu's background color. The ForeColor property specifies the menu's foreground color. The following screen shot shows the menu using the exMenuNormal option:



10250 4 8/8/1994 9/5/1994

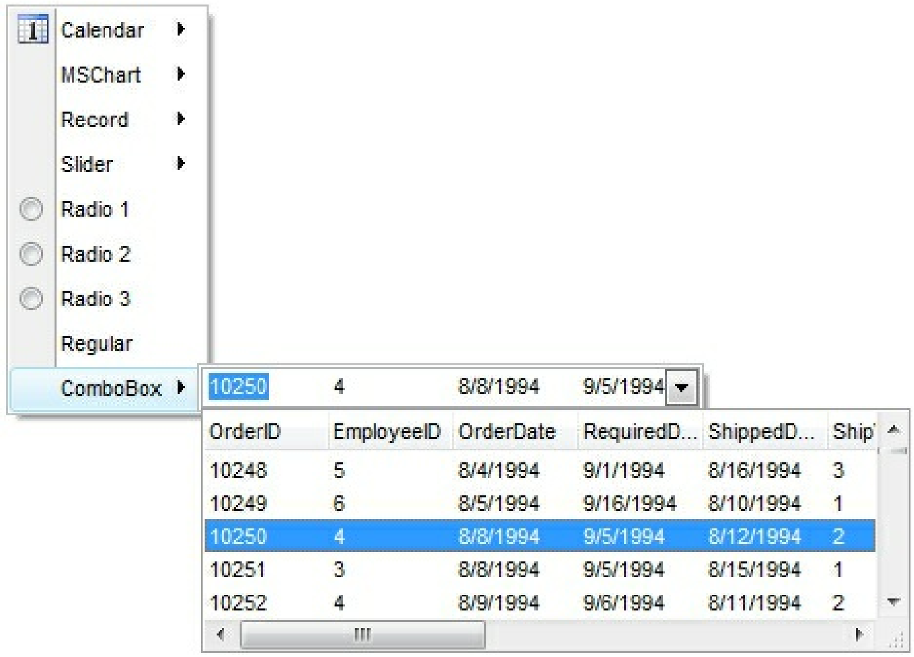
OrderID	EmployeeID	OrderDate	RequiredD...	ShippedD...	Ship
10248	5	8/4/1994	9/1/1994	8/16/1994	3
10249	6	8/5/1994	9/16/1994	8/10/1994	1
10250	4	8/8/1994	9/5/1994	8/12/1994	2
10251	3	8/8/1994	9/5/1994	8/15/1994	1
10252	4	8/9/1994	9/6/1994	8/11/1994	2

</td></tr>
<tr><td>exMenuFlat</td><td>1</td><td>

The BackColor property specifies the menu's background color. The ForeColor property specifies the menu's foreground color. The Background(exMenuFlatLineColor) property indicates the color of line that divides the left to right side of the menu. The FlatBackColor property indicates the color to show the left part of the menu. Use the FlatImageWidth property to specify the width of the column that displays icons/images/check or radio buttons.

You can use this option to show all images, check boxes, radio buttons aligned to the left side as shown in the following screen shot:

</td></tr>
</table>

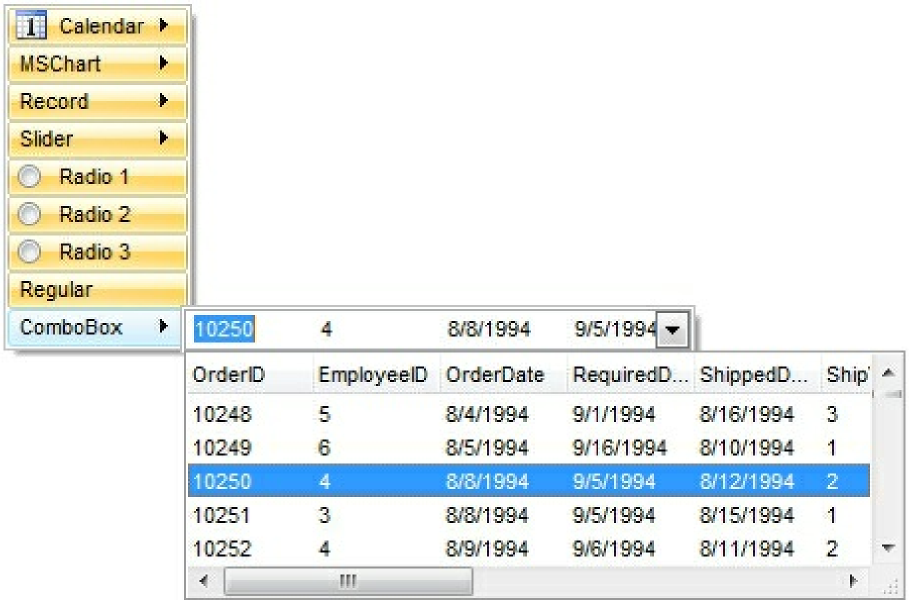


10250 4 8/8/1994 9/5/1994

OrderID	EmployeeID	OrderDate	RequiredD...	ShippedD...	Ship'
10248	5	8/4/1994	9/1/1994	8/16/1994	3
10249	6	8/5/1994	9/16/1994	8/10/1994	1
10250	4	8/8/1994	9/5/1994	8/12/1994	2
10251	3	8/8/1994	9/5/1994	8/15/1994	1
10252	4	8/9/1994	9/6/1994	8/11/1994	2

exMenuButton 2

The BackColor property specifies the menu's background color. The ForeColor property specifies the menu's foreground color. The Background(exMenuButtonItem) property indicates the visual appearance for items in the menu control. You can use this option to apply a new appearance for all items in the control by specifying the The Background(exMenuButtonItem) property to reffer an EBN object link in the following screen shot:



10250 4 8/8/1994 9/5/1994

OrderID	EmployeeID	OrderDate	RequiredD...	ShippedD...	Ship'
10248	5	8/4/1994	9/1/1994	8/16/1994	3
10249	6	8/5/1994	9/16/1994	8/10/1994	1
10250	4	8/8/1994	9/5/1994	8/12/1994	2
10251	3	8/8/1994	9/5/1994	8/15/1994	1
10252	4	8/9/1994	9/6/1994	8/11/1994	2

constants MenuBorderEnum

The MenuBorderEnum indicates the type of the borders to be shown on the menu. The MenuBorderEnum type supports the following values:

Name	Value	Description
NoBorder	0	NoBorder
FlatBorder	1	FlatBorder
SunkenBorder	2	SunkenBorder
RaisedBorder	3	RaisedBorder
EtchedBorder	4	EtchedBorder
BumpBorder	5	BumpBorder
ShadowBorder	6	ShadowBorder
InsetBorder	7	InsetBorder
SingleBorder	8	SingleBorder

constants MenuItemTypeEnum

The MenuItemTypeEnum type specifies the type of Item objects to be collected using the Get method. The Get method can be used to get a collection / safe array of Item objects with a specified characteristics. For instance, you can collect the items of Edit type, or items that holds an Edit field inside. The MenuItemTypeEnum type supports the following values:

Name	Value	Description
exRegularMenuItem	0	Indicates a regular item. A regular item contains no sub-menus, no sub-control, no check box, no radio buttons and it is not a separator item.
exCheckBoxMenuItem	1	Indicates an item with a check box. The Check property specifies whether the item displays a check box.
exRadioButtonMenuItem	2	Indicates an item with a radio button. The Radio property specifies whether the item displays a radio button.
exSubMenuItem	3	Indicates a sub-menu item or an item that displays another menu.
exSeparatorMenuItem	4	Specifies a separator item. The SubMenu property gives access to the items collection being shown in the popup menu.
exControlMenuItem	5	Indicates a sub-menu item that displays an ActiveX or a Window inside. The SubControl property gives access to the Control object being displayed.
exAllMenuItems	15	Indicates any type of item
exImageMenuItem	16	This flag can be combined with any other flag and value to specify the items that display icons using the Image property.
exDisplayTFIMenuItem	32	Reserved.
exEditMenuItem	64	This flag can be combined with any other flag and value to specify the items that display any Edit field inside. The EditCaption property specifies the caption to be shown in the item's edit field.
exDisabledMenuItem	128	This flag can be combined with any other flag and value to specify the disabled items, or items with the Enabled property on False.
		This flag can be combined with any other flag and

exUncheckedMenuItem	256	value to specify un-checked items, or items with the Checked property on False.
exCheckedMenuItem	512	This flag can be combined with any other flag and value to specify checked items, or items with the Checked property on True.
exPartialCheckedMenuItem	768	Reserved.

constants IncrementalSearchEnum

"In computing, incremental search, incremental find or real-time suggestions is a user interface interaction method to progressively search for and filter through text. As the user types text, one or more possible matches for the text are found and immediately presented to the user. ". The IncrementalSearchEnum type specifies how the control performs the incremental searching while user types characters. The IncrementalSearch property the control's incremental search type.

Name	Value	Description
exNoIncrementalSearch	0	No incremental search is performed, when the user types characters.
exISearchStartWith	1	Specifies that the control looks for objects that starts with typed characters, with highlighting the found result.
exISearchContains	2	Specifies that the control looks for objects that contains typed characters, with highlighting the found result.
exISearchFilterFor	16	The control displays just the items that match the typed characters.

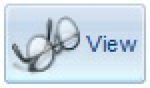
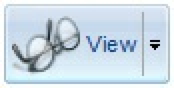
constants PictureDisplayEnum

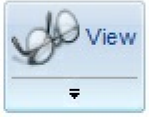
Specifies how a picture object is displayed.

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

constants ShowAsButtonEnum

The ShowAsButtonEnum type specifies the way a button is shown. The ShowAsButton property specifies whether the current item is shown a button or a select button. The ShowAsButtonEnum type supports the following values:

Name	Value	Description
exShowAsButtonNone	0	No button is associated with the current item.
exShowAsButton	1	A button is associated with the current item. This flag can be combined with exShowAsButtonAutoSize flag. 
exShowAsButtonAutoSize	2	The size of the button's caption fits the item's caption. This flag can be combined with any other flags.
exShowAsSelectButton	17	A select button is associated with the current item (show the drop-down button to the right). The exShowAsSelectButton flag has effect only if the current item is a popup item, so it contains sub-items. The SubMenu property gives access to the item's sub menu, while it is a popup item. The Add(Caption,SubMenu) adds a popup item. The item's SubMenu is shown bellow the button, when user clicks the associated arrow. The popup being shown is a local popup, so clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus. 
		A select button is associated with the current item (show the drop-down button to the bottom). The exShowAsSelectButtonBottom flag has effect only if the current item is a popup item, so it contains sub-items. The SubMenu property gives access to the item's sub menu, while it is a popup item. The Add(Caption, SubMenu) adds a popup item (an

exShowAsSelectButtonBottom	273	item that contains sub-items). The item's SubMenu is shown bellow the button, when user clicks the associated arrow. The popup being shown is a local popup, so clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus. 

constants ShowCheckedAsSelectedEnum

The ShowCheckedAsSelected property specifies whether the checked items (all) shows as selected. The ShowCheckedAsSelected property of the Item object specifies whether the individual checked item is shown as selected. The ShowCheckedAsSelectedEnum type supports the following values.

Name	Value	Description
exDisplayItemCheckDefault	0	No highlighting is applied to item.
exDisplayItemCheckHighlight	-1	Highlights or un-highlights the checked/unchecked item, but still the check/radio buttons are shown.
exDisplayItemHighlight	1	Highlights or un-highlights the checked/unchecked item, but check/radio buttons are hidden.
exDisplayItemCheckInherit	2	Inherits the value of the control's ShowCheckedAsSelected property.

constants ShowPopupArrowEnum

The ShowPopupArrowEnum type specifies how the arrow of an item that displays a sub-menu is shown. The ShowPopupArrow specifies how the arrow of an item that displays a sub-menu is shown. The ShowPopupArrowEnum supports the following values:

Name	Value	Description
exPopupArrowNone	0	No arrow is shown.
exShowPopupArrowLight	1	The item shows a light arrow when it displays a sub-menu.
exShowPopupArrowDark	2	The item shows a dark arrow when it displays a sub-menu.

constants ShowPopupEffectEnum

The ShowPopupEffectEnum value indicates the effect to be shown, when the user clicks an item with a sub-menu associated. The ShowPopupEffect property indicates the effect to be applied when the popup menu is shown. The ShowPopupEffectEnum type supports the following values:

Name	Value	Description
exShowPopupDirect	0	The popup menu is shown directly, with no effect.
exShowPopupScroll	-1	The popup menu is scrolling before showing.
exShowPopupLightUp	1	The popup menu is lightning up before showing.

constants SubMenuSortOrderEnum

The SubMenuSortOrderEnum type specifies the way the submenu displays the items. The SortOrder property specifies the sort order to display the items in the sub menu. The SubMenuSortOrderEnum type supports the following values:

Name	Value	Description
exSubMenuUnsorted	0	No sort is applied when the submenu is displayed.
exSubMenuAscending	1	The items in the submenu are alphabetically displayed in ascending order.
exSubMenuDescending	2	The items in the submenu are alphabetically displayed in descending order.
exSubMenuReverse	3	The items in the submenu are displayed in reverse order.

constants ItemTypeEnum

The ItemTypeEnum type specifies the type of items to be added to the control. The ItemType parameter of the Add method specifies the type of the item to be added to the Items collection. The ItemTypeEnum type supports the following values:

Name	Value	Description
Regular	0	Specifies a regular item, with no sub menu.
Separator	1	Specifies a separator item. The Background(exMenuSeparatorItem) property specifies the visual appearance of the separator items.
SubMenu	2	Specifies a sub menu. The SubMenu property gets a collection of Item objects to be displayed on the sub-menu.
SubControl	3	Specifies a popup menu that hosts an ActiveX control or a Window. The SubControl property gets access to the Control object that holds information about the inside ActiveX or Window hosted by the item

constants UIVisualThemeEnum

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

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

Appearance object

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

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

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

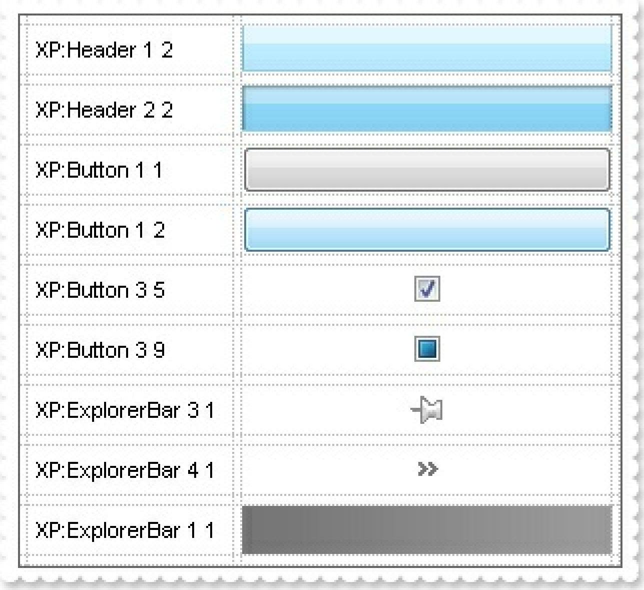
Adds or replaces a skin object to the control.

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

Skin as Variant

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:



XP:Header 1 2

XP:Header 2 2

XP:Button 1 1

XP:Button 1 2

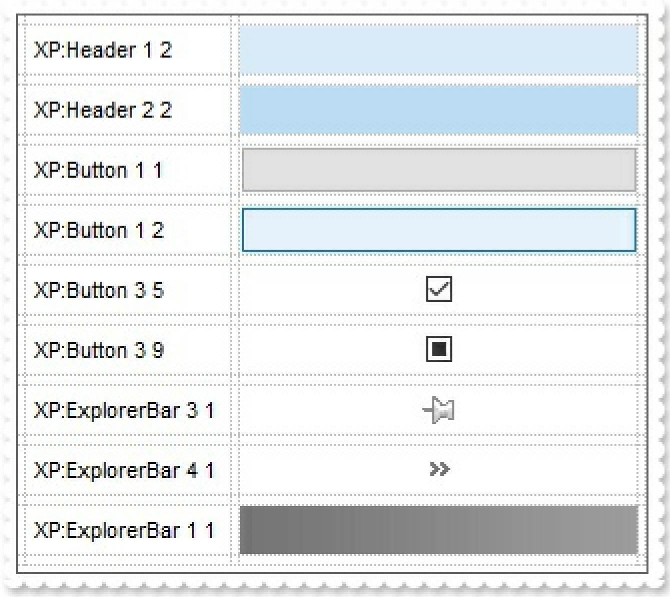
XP:Button 3 5

XP:Button 3 9

XP:ExplorerBar 3 1

XP:ExplorerBar 4 1

XP:ExplorerBar 1 1



XP:Header 1 2

XP:Header 2 2

XP:Button 1 1

XP:Button 1 2

XP:Button 3 5

XP:Button 3 9

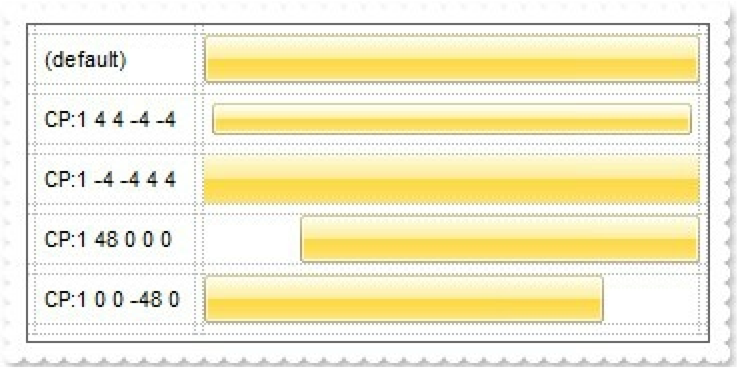
XP:ExplorerBar 3 1

XP:ExplorerBar 4 1

XP:ExplorerBar 1 1

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

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



(default)

CP:1 4 4 -4 -4

CP:1 -4 -4 4 4

CP:1 48 0 0 0

CP:1 0 0 -48 0

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

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

The identifier you choose for the skin is very important to be used in the background properties like explained bellow. Shortly, the color properties uses 4 bytes (DWORD, double WORD, and so on) to hold a RGB value. More than that, the first byte (most significant byte in the color) is used only to specify system color. if the first bit in the byte is

1, the rest of bits indicates the index of the system color being used. So, we use the last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. So, since the 7 bits can cover 127 values, excluding 0, we have 126 possibilities to store an identifier in that byte. This way, a DWORD expression indicates the background color stored in RRGGBB format and the index of the skin (ID parameter) in the last 7 bits in the high significant byte of the color. For instance, the BackColor = BackColor Or &H2000000 indicates that we apply the skin with the index 2 using the old color, to the object that BackColor is applied.

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

Control/ClassName	Part	States
BUTTON	BP_CHECKBOX = 3	CBS_UNCHECKED 1 CBS_UNCHECKE CBS_UNCHECKED = 3 CBS_UNCHECKED = 4 CBS_CHECKED 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRESS CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
	BP_PUSHBUTTON = 1	PBS_NORMAL = 1 = 2 PBS_PRESSED PBS_DISABLED = PBS_DEFAULTED =
	BP_RADIOBUTTON = 2	RBS_UNCHECKED 1 RBS_UNCHECKE RBS_UNCHECKED = 3 RBS_UNCHECKED = 4 RBS_CHECKED 5 RBS_CHECKEDH RBS_CHECKEDPR RBS_CHECKEDDIS
	BP_USERBUTTON = 5	

CLOCK	CLP_TIME = 1	CLS_NORMAL = 1
COMBOBOX	CP_DROPDOWNBUTTON = 1	CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED =
EDIT	EP_CARET = 2	

	EP_EDITTEXT = 1	ETS_NORMAL = 1
		2 ETS_SELECTED
		ETS_DISABLED = 4
		ETS_FOCUSED = 5
		ETS_READONLY =
		ETS_ASSIST = 7

EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
	EBP_HEADERCLOSE = 2	EBHC_NORMAL = EBHC_HOT = 2 EBHC_PRESSED =
	EBP_HEADERPIN = 3	EBHP_NORMAL = EBHP_HOT = 2 EBHP_PRESSED = EBHP_SELECTEDN 4 EBHP_SELECTEI EBHP_SELECTEDF 6
	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSEI
	EBP_NORMALGROUPBACKGROUND = 5	
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_NORMAL = EBNGC_HOT = 2 EBNGC_PRESSED
	EBP_NORMALGROUPEXPAND = 7	EBNGE_NORMAL = EBNGE_HOT = 2 EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
	EBP_SPECIALGROUPCOLLAPSE = 10	EBSGC_NORMAL = EBSGC_HOT = 2 EBSGC_PRESSED
	EBP_SPECIALGROUPEXPAND = 11	EBSGE_NORMAL = EBSGE_HOT = 2 EBSGE_PRESSED

	EBP_SPECIALGROUPHEAD = 12	
HEADER	HP_HEADERITEM = 1	HIS_NORMAL = 1 H 2 HIS_PRESSED =
	HP_HEADERITEMLEFT = 2	HILS_NORMAL = 1 = 2 HILS_PRESSE
	HP_HEADERITEMRIGHT = 3	HIRS_NORMAL = 1 = 2 HIRS_PRESSE
	HP_HEADERSORTARROW = 4	HSAS_SORTEDUP HSAS_SORTEDDO
LISTVIEW	LVP_EMPTYTEXT = 5	
	LVP_LISTDETAIL = 3	
	LVP_LISTGROUP = 2	
	LVP_LISTITEM = 1	LIS_NORMAL = 1 L 2 LIS_SELECTED = LIS_DISABLED = 4 LIS_SELECTEDNO 5
	LVP_LISTSORTEDDETAIL = 4	

MENU	MP_MENUBARDROPDOWN = 4	MS_NORMAL = 1
		MS_SELECTED = 2
		MS_DEMOTED = 3

	MP_MENUBARITEM = 3	MS_NORMAL = 1
		MS_SELECTED = 2
		MS_DEMOTED = 3

	MP_CHEVRON = 5	MS_NORMAL = 1
		MS_SELECTED = 2
		MS_DEMOTED = 3

	MP_MENUDROPDOWN = 2	MS_NORMAL = 1
		MS_SELECTED = 2
		MS_DEMOTED = 3

	MP_MENUITEM = 1	MS_NORMAL = 1
		MS_SELECTED = 2
		MS_DEMOTED = 3

	MP_SEPARATOR = 6	MS_NORMAL = 1
		MS_SELECTED = 2
		MS_DEMOTED = 3

MENUBAND	MDP_NEWAPPBUTTON = 1	MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_DISABLED = MDS_CHECKED =

		MDS_HOTCHECKE
	MDP_SEPERATOR = 2	
PAGE	PGRP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	PGRP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	PGRP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	PGRP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
PROGRESS	PP_BAR = 1	
	PP_BARVERT = 2	
	PP_CHUNK = 3	
	PP_CHUNKVERT = 4	
REBAR	RP_BAND = 3	
	RP_CHEVRON = 4	CHEVS_NORMAL = CHEVS_HOT = 2 CHEVS_PRESSED
	RP_CHEVRONVERT = 5	
	RP_GRIPPER = 1	
	RP_GRIPPERVERT = 2	
SCROLLBAR	SBP_ARROWBTN = 1	ABS_DOWNDISAB ABS_DOWNHOT, ABS_DOWNNORM ABS_DOWNPRESS ABS_UPDISABLED ABS_UPHOT, ABS_UPNORMAL, ABS_UPPRESSED, ABS_LEFTDISABLI ABS_LEFTHOT, ABS_LEFTNORMA ABS_LEFTPRESSE ABS_RIGHTDISAB

		ABS_RIGHTHOT, ABS_RIGHTNORM ABS_RIGHTPRESS
	SBP_GRIPPERHORZ = 8	
	SBP_GRIPPERVERT = 9	
	SBP_LOWERTRACKHORZ = 4	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_LOWERTRACKVERT = 6	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNHORZ = 2	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNVERT = 3	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKHORZ = 5	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKVERT = 7	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_SIZEBOX = 10	SZB_RIGHTALIGN SZB_LEFTALIGN =
SPIN	SPNP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	SPNP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	SPNP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =

	SPNP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
STARTPANEL	SPP_LOGOFF = 8	
	SPP_LOGOFFBUTTONS = 9	SPLS_NORMAL = SPLS_HOT = 2 SPLS_PRESSED =
	SPP_MOREPROGRAMS = 2	
	SPP_MOREPROGRAMSARROW = 3	SPS_NORMAL = 1 = 2 SPS_PRESSED
	SPP_PLACESLIST = 6	
	SPP_PLACESLISTSEPARATOR = 7	
	SPP_PREVIEW = 11	
	SPP_PROGLIST = 4	
	SPP_PROGLISTSEPARATOR = 5	
	SPP_USERPANE = 1	
	SPP_USERPICTURE = 10	
STATUS	SP_GRIPPER = 3	
	SP_PANE = 1	
	SP_GRIPPERPANE = 2	
TAB	TABP_BODY = 10	
	TABP_PANE = 9	

	TABP_TABITEM = 1	TIS_NORMAL = 1
		2 TIS_SELECTED =
		TIS_DISABLED = 4
		TIS_FOCUSED = 5

	TABP_TABITEMBOTHEDGE = 4	TIBES_NORMAL = TIBES_HOT = 2 TIBES_SELECTED TIBES_DISABLED TIBES_FOCUSED =
	TABP_TABITEMLEFTEDGE = 2	TILES_NORMAL = TILES_HOT = 2 TILES_SELECTED TILES_DISABLED = TILES_FOCUSED =
		TIRES_NORMAL = TIRES_HOT = 2

	TABP_TABITEMRIGHTEDGE = 3	TIRES_SELECTED TIRES_DISABLED TIRES_FOCUSED
	TABP_TOPTABITEM = 5	TTIS_NORMAL = 1 = 2 TTIS_SELECTE TTIS_DISABLED = TTIS_FOCUSED =
	TABP_TOPTABITEMBOTHEDGE = 8	TTIBES_NORMAL TTIBES_HOT = 2 TTIBES_SELECTE TTIBES_DISABLED TTIBES_FOCUSED
	TABP_TOPTABITEMLEFTEDGE = 6	TTILES_NORMAL TTILES_HOT = 2 TTILES_SELECTED TTILES_DISABLED TTILES_FOCUSED
	TABP_TOPTABITEMRIGHTEDGE = 7	TTIRES_NORMAL TTIRES_HOT = 2 TTIRES_SELECTE TTIRES_DISABLED TTIRES_FOCUSED
TASKBAND	TDP_GROUPCOUNT = 1	
	TDP_FLASHBUTTON = 2	
	TDP_FLASHBUTTONGROUPMENU = 3	
TASKBAR	TBP_BACKGROUNDBOTTOM = 1	
	TBP_BACKGROUNDLEFT = 4	
	TBP_BACKGROUNDRIGHT = 2	
	TBP_BACKGROUNDTOP = 3	
	TBP_SIZINGBARBOTTOM = 5	
	TBP_SIZINGBARBOTTOMLEFT = 8	
	TBP_SIZINGBARRIGHT = 6	
	TBP_SIZINGBARTOP = 7	

TOOLBAR	TP_BUTTON = 1	TS_NORMAL = 1 T
		TS_PRESSED = 3
		TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED

		TS_NORMAL = 1 T
		TS_PRESSED = 3

	TP_DROPDOWNBUTTON = 2	TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED

	TP_SPLITBUTTON = 3	TS_NORMAL = 1 T
		TS_PRESSED = 3
		TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED

	TP_SPLITBUTTONDROPDOWN = 4	TS_NORMAL = 1 T
		TS_PRESSED = 3
		TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED

	TP_SEPARATOR = 5	TS_NORMAL = 1 T
		TS_PRESSED = 3
		TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED

	TP_SEPARATORVERT = 6	TS_NORMAL = 1 T
		TS_PRESSED = 3
		TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED

TOOLTIP	TTP_BALLOON = 3	TTBS_NORMAL = TTBS_LINK = 2
	TTP_BALLOONTITLE = 4	TTBS_NORMAL = TTBS_LINK = 2
	TTP_CLOSE = 5	TTCS_NORMAL = TTCS_HOT = 2 TTCS_PRESSED =
	TTP_STANDARD = 1	TTSS_NORMAL = TTSS_LINK = 2
	TTP_STANDARDTITLE = 2	TTSS_NORMAL = TTSS_LINK = 2
TRACKBAR	TKP_THUMB = 3	TUS_NORMAL = 1 2 TUS_PRESSED = TUS_FOCUSED = TUS_DISABLED =
	TKP_THUMBBOTTOM = 4	TUBS_NORMAL = TUBS_HOT = 2 TUBS_PRESSED =

		TUBS_FOCUSED = TUBS_DISABLED =
	TKP_THUMBLEFT = 7	TUVLS_NORMAL = TUVLS_HOT = 2 TUVLS_PRESSED TUVLS_FOCUSED TUVLS_DISABLED
	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = TUVRS_HOT = 2 TUVRS_PRESSED TUVRS_FOCUSED TUVRS_DISABLED
	TKP_THUMBTOP = 5	TUTS_NORMAL = 1 TUTS_HOT = 2 TUTS_PRESSED = TUTS_FOCUSED = TUTS_DISABLED =
	TKP_THUMBVERT = 6	TUVS_NORMAL = TUVS_HOT = 2 TUVS_PRESSED = TUVS_FOCUSED = TUVS_DISABLED =
	TKP_TICS = 9	TSS_NORMAL = 1
	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	TNP_ANIMBACKGROUND = 2	
	TNP_BACKGROUND = 1	
TREEVIEW	TVP_BRANCH = 3	
	TVP_GLYPH = 2	GLPS_CLOSED = GLPS_OPENED = 2
	TVP_TREEITEM = 1	TREIS_NORMAL = TREIS_HOT = 2 TREIS_SELECTED TREIS_DISABLED TREIS_SELECTED = 5
WINDOW	WP_CAPTION = 1	CS_ACTIVE = 1 CS = 2 CS_DISABLED
	WP_CAPTIONSIZINGTEMPLATE = 30	

WP_CLOSEBUTTON = 18	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_DIALOG = 29	
WP_FRAMEBOTTOM = 9	FS_ACTIVE = 1 FS = 2
WP_FRAMEBOTTOMSIZINGTEMPLATE = 36	
WP_FRAMELEFT = 7	FS_ACTIVE = 1 FS = 2
WP_FRAMELEFTSIZINGTEMPLATE = 32	
WP_FRAMERIGHT = 8	FS_ACTIVE = 1 FS = 2
WP_FRAMERIGHTSIZINGTEMPLATE = 34	
WP_HELPBUTTON = 23	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_HORZSCROLL = 25	HSS_NORMAL = 1 = 2 HSS_PUSHED HSS_DISABLED =
WP_HORZTHUMB = 26	HTS_NORMAL = 1 2 HTS_PUSHED = HTS_DISABLED =
WP_MAX_BUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_MAXCAPTION = 5	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_MDICLOSEBUTTON = 20	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_MDIHELPBUTTON = 24	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_MDIMINBUTTON = 16	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
	RBS_NORMAL = 1

WP_MDIRESTOREBUTTON = 22	= 2 RBS_PUSHED RBS_DISABLED =
WP_MDISYSBUTTON = 14	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_MINBUTTON = 15	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
WP_MINCAPTION = 3	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_RESTOREBUTTON = 21	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLCAPTION = 2	CS_ACTIVE = 1 CS = 2 CS_DISABLED
WP_SMALLCAPTIONSIZINGTEMPLATE = 31	
WP_SMALLCLOSEBUTTON = 19	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_SMALLFRAMEBOTTOM = 12	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37	
WP_SMALLFRAMELEFT = 10	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMELEFTSIZINGTEMPLATE = 33	
WP_SMALLFRAMERIGHT = 11	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35	
WP_SMALLHELPBUTTON	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON	MAXBS_NORMAL = MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED

WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_VERTSCROLL = 27	VSS_NORMAL = 1 = 2 VSS_PUSHED VSS_DISABLED =
WP_VERTTHUMB = 28	VTS_NORMAL = 1 2 VTS_PUSHED = VTS_DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

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

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

property Appearance.RenderType as Long

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

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

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

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

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

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

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

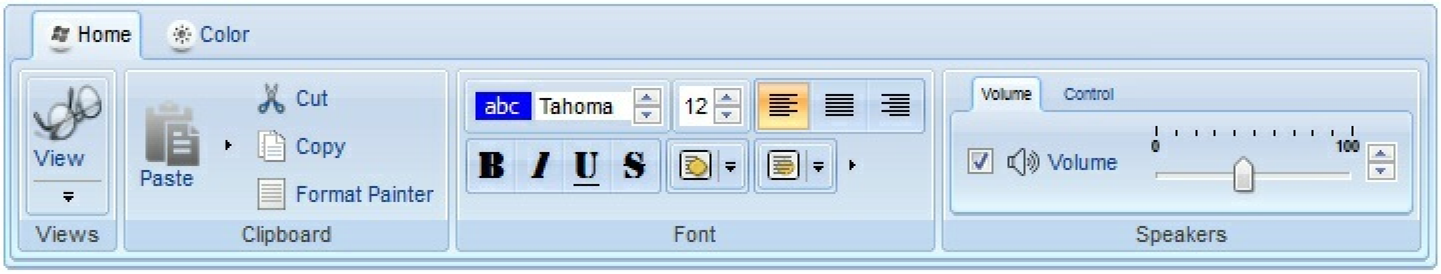
In the following screen shot the following objects displays the current EBN with a different color:

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

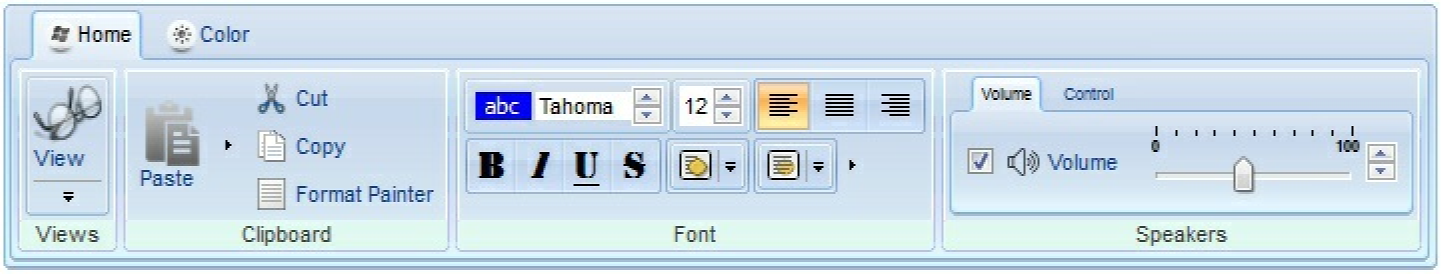
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

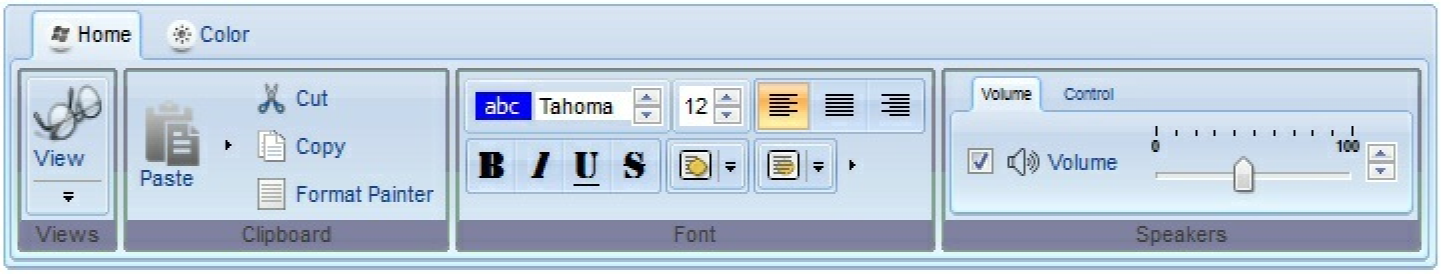
- ***-3**, no color is applied*. For instance, the Property = &H1FF0000 is displayed as would be .Property = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



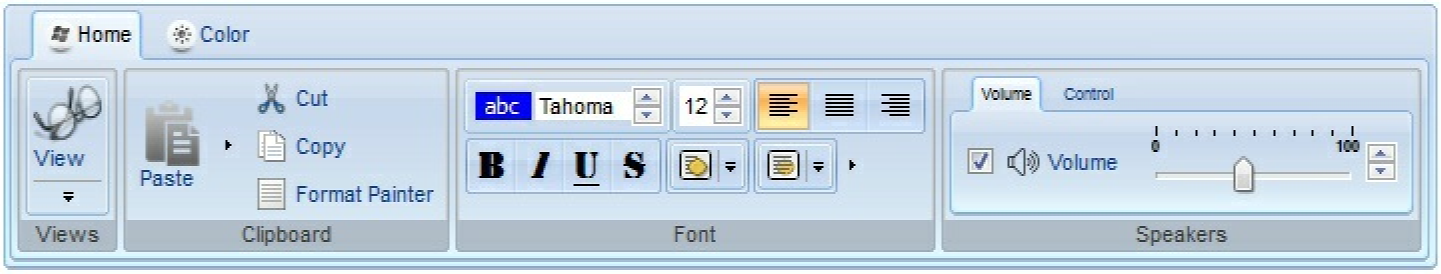
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the Property = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the part 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 Property = &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), ...)

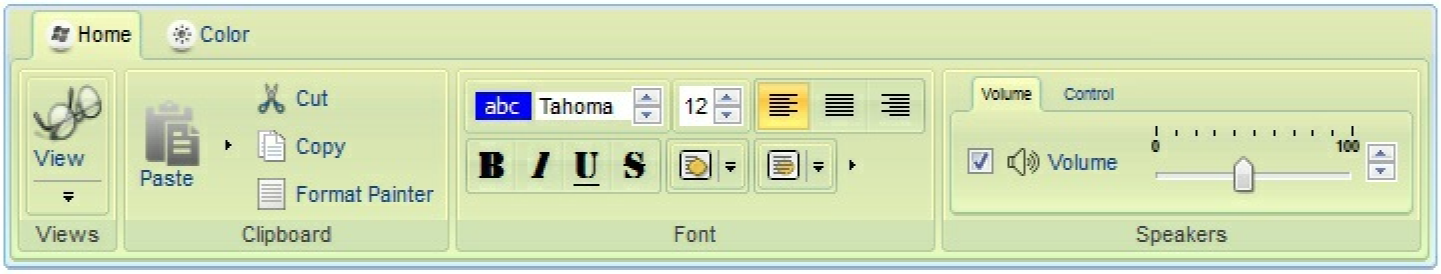


- ***0**, default*, the specified color is applied to the EBN. For instance, the Property = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

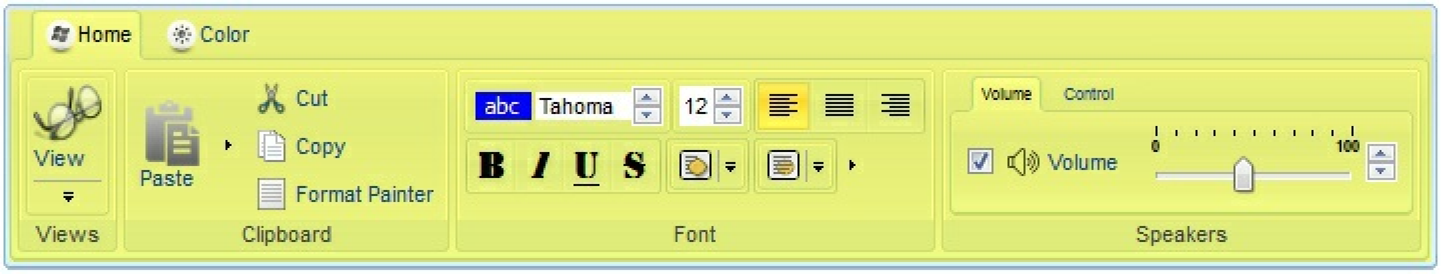


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

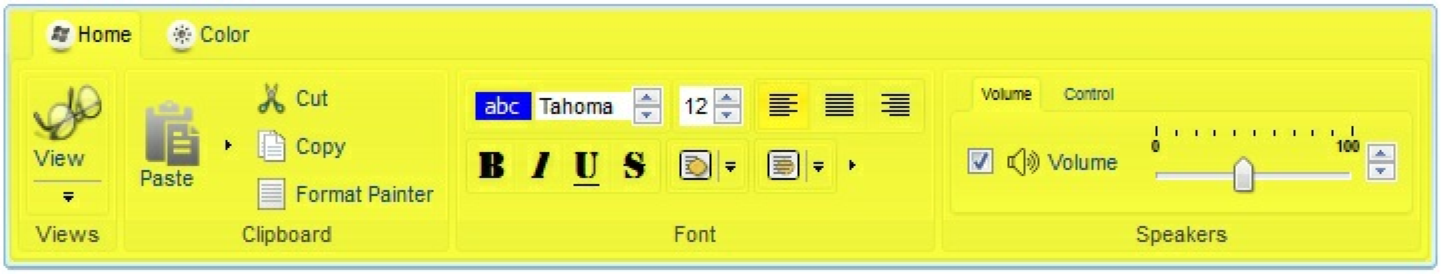
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



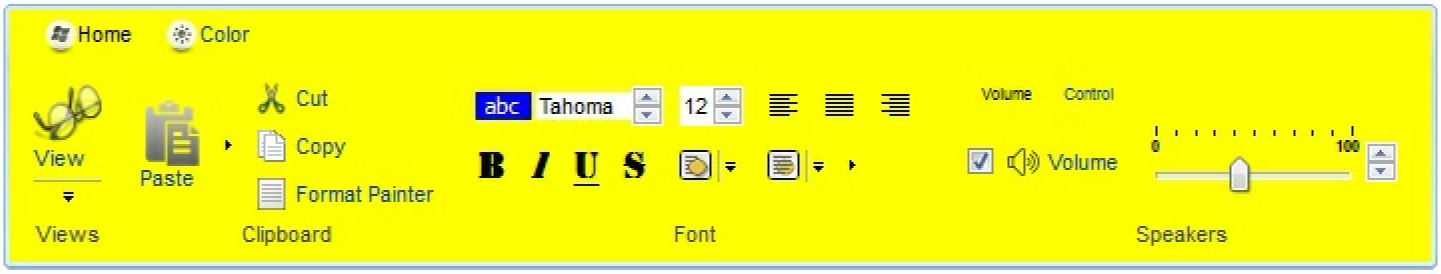
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



*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):*



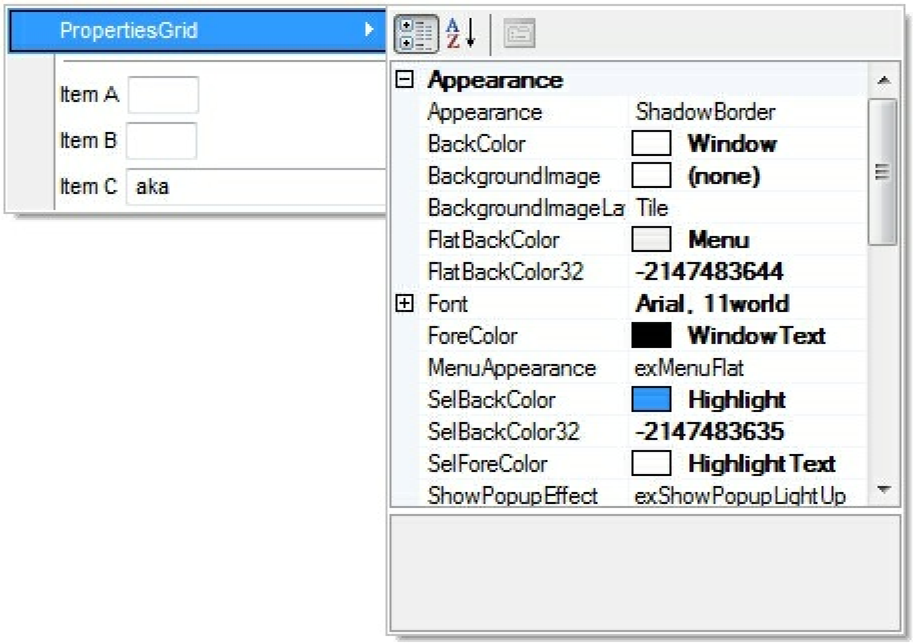
*The following picture shows the control with the RenderType property on 0x**FF**00FFFF (100% Yellow, 0xFF or 255 in decimal is 100% from 255):*



Control object

The Control object holds properties to access the ActiveX or Window to be hosted by the item. The eXContextMenu component can host any ActiveX control or an already created window. The Control object can be accessed through the SubControl property of the Item object.

The following screen shot shows the eXContextMenu/NET that hosts a PropertyGrid control:



⊟ **Appearance**	
Appearance	ShadowBorder
BackColor	□ **Window**
BackgroundImage	□ **(none)**
BackgroundImageLa	Tile
FlatBackColor	□ **Menu**
FlatBackColor32	**-2147483644**
⊞ Font	**Arial, 11world**
ForeColor	■ **WindowText**
MenuAppearance	exMenuFlat
SelBackColor	🟦 **Highlight**
SelBackColor32	**-2147483635**
SelForeColor	□ **HighlightText**
ShowPopupEffect	exShowPopupLightUp

The Control object supports the following properties and methods:

Name	Description
CloseOn	Indicates when the control is closed.
ControlID	Specifies the control's identifier.
Create	Creates the component.
Height	Specifies the control's height.
LicenseKey	Specifies the control's runtime license key.
Object	Gets the object.
Width	Specifies the control's width.
Window	Specifies the handle of the window to be hosted.

property Control.CloseOn as CloseOnEnum

Indicates when the control is closed.

Type	Description
CloseOnEnum	A CloseOnEnum expression that specifies when the context menu is closed once the user clicks the inside ActiveX control.

By default, the CloseOn property is exUser. In other words, the context menu is not closed once the user clicks the ActiveX control.

property Control.ControlID as String

Specifies the control's identifier.

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

The ControlID and LicenseKey properties must be set before calling Create method. The Create method creates an ActiveX control given its identifier and its runtime license key, if required. A control identifier, or programmatic identifier, is a registry entry that can be associated with a CLSID. The format of a control identifier is <*Vendor*>.<*Component*>.<*Version*>, separated by periods and with no spaces, as in Word.Document.6.

For instance, the control's identifier for Microsoft Calendar Control is "MSCAL.Calendar", the control's identifier for Exontrol ExGrid Control is "Exontrol.Grid", and so on.

method Control.Create ()

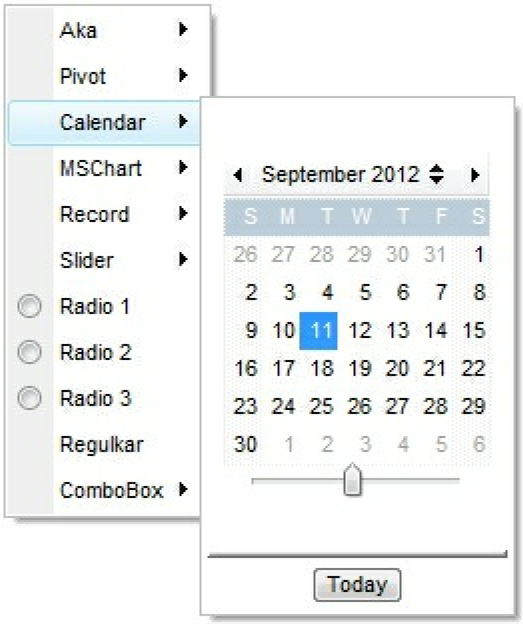
Creates the component.

Type	Description

The Create method creates the ActiveX control. The Create method creates the control based on its control's identifier. Use the ControlID and LicenseKey properties to specify the control's identifier and the runtime license key for the control, if required (please make sure that the runtime license key is not identical with your development license key). If the Create method fails, the Object property gets nothing. Use the Object property to access the ActiveX control's properties and methods. Use the CloseOn property to specify how the item that hosts an ActiveX control is closed using the mouse. Use the Width and Height properties to specify the size of the item that hosts the ActiveX control. The control fires the OleEvent event when an inside ActiveX control fires an event. The look and feel of the inner ActiveX control depends on the identifier you are using, and the version of the library that implements the ActiveX control, so you need to consult the documentation of the inner ActiveX control you are inserting inside the eXMenu control.

In case you are using the /NET assembly version, you can use the Window property to assign a Window/Control to an Item.

The following screen shot displays an item with an ExCalendar inside:



The following samples shows how to load an ActiveX control (Exontrol.Calendar)

VB6,VBA (MS Access, Excell...),VB.NET for /COM

```
With CreateObject("Exontrol.ContextMenu")
   With .Items.Add("Calendar",3).SubControl
```

```
		.ControlID = "Exontrol.Calendar"
		.Create
	End With
	.Select
End With
```

VB.NET

```
' Add 'exontrol.excontextmenu.dll' reference to your project.
With New exontrol.EXCONTEXTMENULib.excontextmenu()
	With .Items.Add("Calendar",3).SubControl
		.ControlID = "Exontrol.Calendar"
		.Create()
	End With
	.Select()
End With
```

C++

```
/*
	Includes the definition for CreateObject function like follows:
	#include <comdef.h>
	IUnknownPtr CreateObject( BSTR Object )
	{
		IUnknownPtr spResult;
		spResult.CreateInstance( Object );
		return spResult;
	};
*/
/*
	Copy and paste the following directives to your header file as
	it defines the namespace 'EXCONTEXTMENULib' for the library: 'ExContextMenu
1.0 Type Library'
	#import <ExContextMenu.dll>
	using namespace EXCONTEXTMENULib;
*/
EXCONTEXTMENULib::IExContextMenuPtr var_ExContextMenu =
::CreateObject(L"Exontrol.ContextMenu");
```

```
	EXCONTEXTMENULib::IControlPtr var_Control = var_ExContextMenu->GetItems()->Add(L"Calendar",long(3),vtMissing)->GetSubControl();
		var_Control->PutControlID(L"Exontrol.Calendar");
		var_Control->Create();
	var_ExContextMenu->Select(vtMissing,vtMissing,vtMissing);
```

C++ Builder

```
/*
	Select the Component\Import Component...\Import a Type Library,
	to import the following Type Library:
		ExContextMenu 1.0 Type Library
	TypeLib: e:\Exontrol\ExContextMenu\project\Site\ExContextMenu.dll
	to define the namespace: Excontextmenulib_tlb
*/
//#include "EXCONTEXTMENULIB_TLB.h"
Excontextmenulib_tlb::IExContextMenuPtr var_ExContextMenu = Variant::CreateObject(L"Exontrol.ContextMenu");
	Excontextmenulib_tlb::IControlPtr var_Control = var_ExContextMenu->Items->Add(L"Calendar",TVariant(3),TNoParam())->SubControl;
		var_Control->ControlID = L"Exontrol.Calendar";
		var_Control->Create();
	var_ExContextMenu->Select(TNoParam(),TNoParam(),TNoParam());
```

C#

```
// Add 'exontrol.excontextmenu.dll' reference to your project.
exontrol.EXCONTEXTMENULib.excontextmenu var_ExContextMenu = new exontrol.EXCONTEXTMENULib.excontextmenu();
	exontrol.EXCONTEXTMENULib.Control var_Control = var_ExContextMenu.Items.Add("Calendar",3,null).SubControl;
		var_Control.ControlID = "Exontrol.Calendar";
		var_Control.Create();
	var_ExContextMenu.Select(null,null,null);
```

C# for /COM

```
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
```

```
EXCONTEXTMENULib.ExContextMenu var_ExContextMenu = new
EXCONTEXTMENULib.ExContextMenu();
   EXCONTEXTMENULib.Control var_Control =
var_ExContextMenu.Items.Add("Calendar",3,null).SubControl;
      var_Control.ControlID = "Exontrol.Calendar";
      var_Control.Create();
   var_ExContextMenu.Select(null,null,null);
```

X++ (Dynamics Ax 2009)

```
COM com_Control,com_ExContextMenu,com_Items,com_item;
anytype var_Control,var_ExContextMenu,var_Items,var_item;
;
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
var_ExContextMenu = COM::createFromObject(new
EXCONTEXTMENULib.excontextmenu()); com_ExContextMenu = var_ExContextMenu;
   var_Items = COM::createFromObject(com_ExContextMenu.Items()); com_Items =
var_Items;
   var_item =
COM::createFromObject(com_Items).Add("Calendar",COMVariant::createFromInt(3));
com_item = var_item;
   var_Control = com_item.SubControl(); com_Control = var_Control;
      com_Control.ControlID("Exontrol.Calendar");
      com_Control.Create();
   com_ExContextMenu.Select();
```

Delphi 8 (.NET only)

```
with (ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMenu'))
as EXCONTEXTMENULib.ExContextMenu) do
begin
   with Items.Add('Calendar',TObject(3),Nil).SubControl do
   begin
      ControlID := 'Exontrol.Calendar';
      Create();
   end;
   Select(Nil,Nil,Nil);
end;
```

Delphi (standard)

```
with
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMen
 as EXCONTEXTMENULib_TLB.ExContextMenu) do
begin
	with Items.Add('Calendar',OleVariant(3),Null).SubControl do
	begin
		ControlID := 'Exontrol.Calendar';
		Create();
	end;
	Select(Null,Null,Null);
end;
```

VFP

```
with CreateObject("Exontrol.ContextMenu")
	with .Items.Add("Calendar",3).SubControl
		.ControlID = "Exontrol.Calendar"
		.Create
	endwith
	.Select()
endwith
```

XBasic (Alpha Five)

```
' Occurs when the user presses and then releases the left mouse button over the control.
function Click as v ()
	Dim oPivot as P
	Dim var_Control as P
	Dim var_ExContextMenu as P
	oPivot = topparent:CONTROL_ACTIVEX1.activex
	var_ExContextMenu = OLE.Create("Exontrol.ContextMenu")
		var_Control = var_ExContextMenu.Items.Add("Calendar",3).SubControl
			var_Control.ControlID = "Exontrol.Calendar"
			var_Control.Create()
		var_ExContextMenu.Select()
```

```
end function

Dim oPivot as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
```

Visual Objects

```
local var_ExContextMenu as IExContextMenu
// Generate Source for 'ExContextMenu 1.0 Type Library' server from
Tools\Automation Server...
var_ExContextMenu := IExContextMenu{"Exontrol.ContextMenu"}
    var_Control := var_ExContextMenu:Items:Add("Calendar",3,nil):SubControl
        var_Control:ControlID := "Exontrol.Calendar"
        var_Control:Create()
    var_ExContextMenu:Select(nil,nil,nil)
```

property Control.Height as Long

Specifies the control's height.

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

By default, the Height property is 128 pixels. Use the Height property to specify the height of the inside control. The Height property has effect only if Create method is called after. Use the Width property to specify the control's width.

property Control.LicenseKey as String

Specifies the control's runtime license key.

Type	Description
String	A string expression that indicates the control's runtime license key.

The LicenseKey property must be set only if the control that you are going to use requires a runtime license key. Please contact the vendor of the control to know if the control requires a runtime license key. The control's runtime license key is not identical with your development license key. The LicenseKey property must be set before calling Create method. Please keep in mind that the vendor/provider of the ActiveX control you want to insert to an item is responsible for the control's runtime license key. Exontrol can provide the runtime license key for our components only.

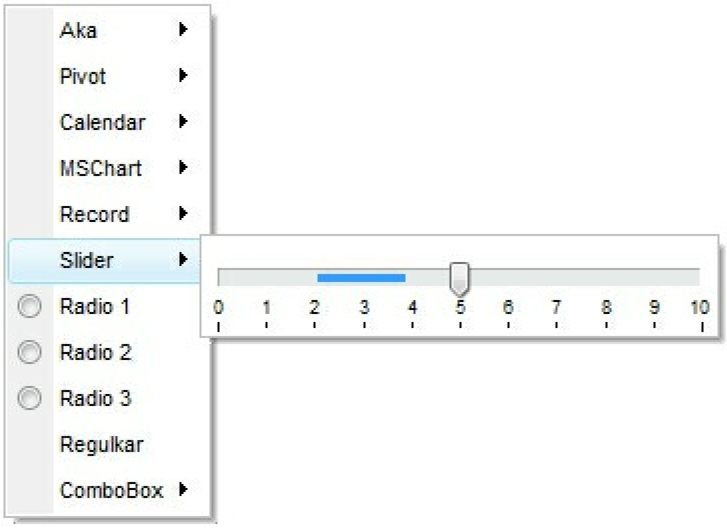
property Control.Object as Object

Gets the object.

Type	Description
Object	An Object created by the Create method.

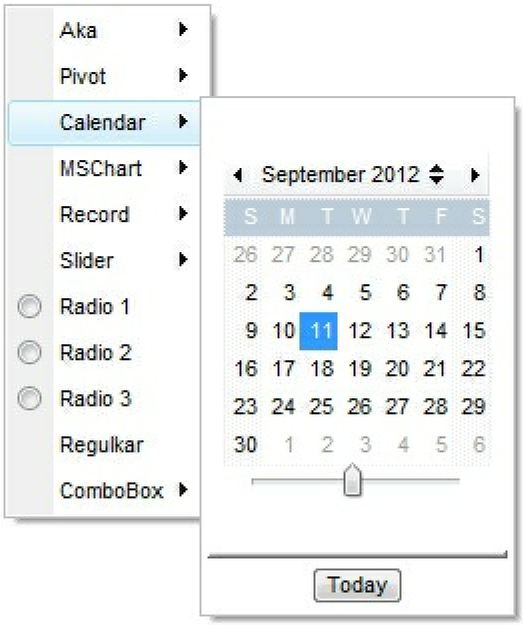
Use the Object property to access to control's properties and methods. The type of the created object depends on ControlID property. The Object property gets nothing if no object was created. Use the Create method to create the inside ActiveX control. The control fires the OleEvent event when an inside ActiveX control fires an event. The look and feel of the inner ActiveX control depends on the identifier you are using, and the version of the library that implements the ActiveX control, so you need to consult the documentation of the inner ActiveX control you are inserting inside the eXMenu control.

The following screen shot displays an item with an ExSlider inside:



3

The following screen shot displays an item with an ExCalendar inside:



property Control.Width as Long

Specifies the control's width.

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

By default, the Width property is 128 pixels. Use the Width property to specify the width of the inside control. The Width property has effect only if Create method is called after. Use the Height property to specify the control's height.

property Control.Window as Variant

Specifies the handle of the window to be hosted.

Type	Description
Variant	A Long expression that specifies the handle of the Window to be hosted by the Item.

Use the Window property to assign a Window to an item. The Window may be used by the /COM or /NET version by providing a valid handle to the window to be shown on the item. The /COM object may use the Create method to create an inside ActiveX control.

The following VB/NET sample displays the form's PropertyGrid control to an Item (/NET version):

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    ' Add 'exontrol.excontextmenu.dll' reference to your project.
    With Excontextmenu1
        With .Items
            With .Add("PropertiesGrid", 3).SubControl
                .Width = 256
                .Height = 312
                .Window = PropertyGrid1
            End With
        End With
    End With

    PropertyGrid1.SelectedObject = Excontextmenu1

End Sub
```

The following C# sample displays the form's PropertyGrid control to an Item (/NET version):

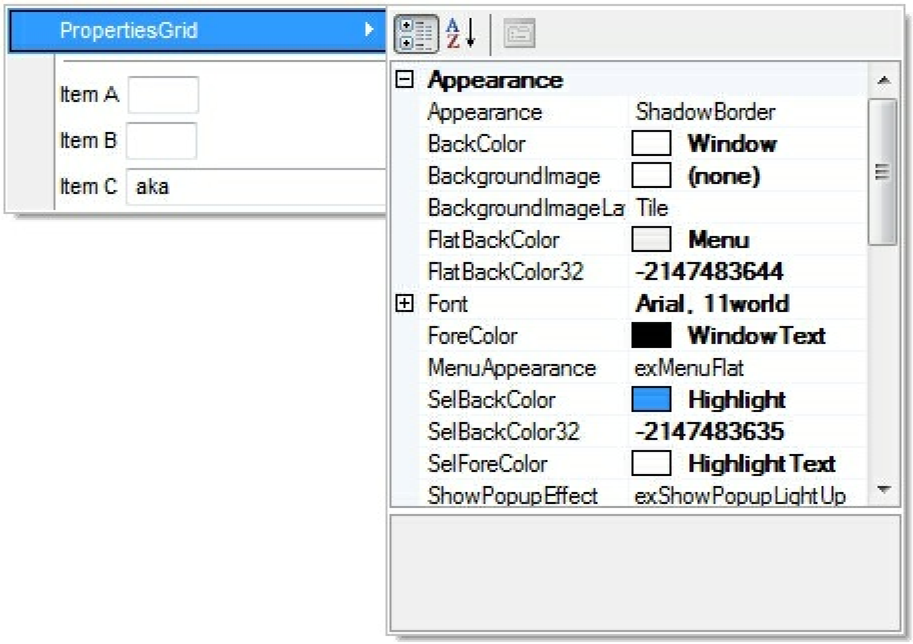
```
private void Form1_Load(object sender, EventArgs e)
{
    exontrol.EXCONTEXTMENULib.Items var_Items = excontextmenu1.Items;
    exontrol.EXCONTEXTMENULib.Control var_Control = var_Items.Add("PropertiesGrid", 3,
null).SubControl;
    var_Control.Width = 256;
    var_Control.Height = 312;
    var_Control.Window = propertyGrid1;

    propertyGrid1.SelectedObject = excontextmenu1;

}
```

ExContextMenu object

The eXContextMenu component displays and handles a context menu (also called contextual, shortcut, and popup or pop-up menu). A context menu is a menu in a graphical user interface (GUI) that appears upon user interaction, such as a right-click mouse operation. The eXContextMenu component is written from scratch, and does NOT use the system's popup menu. For instance, the /NET's System.Windows.Controls.ContextMenu does not support a modal form, so you have to assign a handler for each item, instead the eXContextMenu component waits for the user to make the selection, and returns the selected values. Also another major difference is that the System.Windows.Controls.ContextMenu is closed once any item is clicked, while in the eXContextMenu component, this is not required, so you can check multiple check boxes, and when you click outside, the Select method returns the selected values.



Item C: aka

⊟ Appearance

Appearance	ShadowBorder
BackColor	□ Window
BackgroundImage	□ (none)
BackgroundImageLa	Tile
FlatBackColor	□ Menu
FlatBackColor32	-2147483644
⊞ Font	Arial, 11world
ForeColor	■ WindowText
MenuAppearance	exMenuFlat
SelBackColor	🟦 Highlight
SelBackColor32	-2147483635
SelForeColor	□ HighlightText
ShowPopupEffect	exShowPopupLightUp

The eXContextMenu supports the following properties and methods:

Name	Description
AllowToggleRadio	Allows or prevents toggling the radio state.
AllowToolTip	Allows or prevents showing the item's tooltip.
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.
CloseOnClick	Gets or sets a value that specifies whether the context

	menu is closing.
Cursor	Gets or sets the cursor that is displayed when the mouse pointer hovers the control.
Debug	Retrieves or sets a value that indicating whether the item's identifier is visible.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
FlatBackColor	Specifies the color to left part of the menu.
FlatImageWidth	Specifies the width of the column to display the icons/images when the control's MenuAppearance is exMenuFlat.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
Get	Retrieves an array of Item objects that meet the criteria.
GetChecked	Retrieves an array of Item objects, that displays a check box which is checked.
GetRadio	Retrieves an array of Item objects of radio type in the same group, that are checked.
GetUnchecked	Retrieves an array of Item objects, that displays a check box which is unchecked.
HTMLPicture	Adds or replaces a picture in HTML captions.
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.
IncrementalSearch	Specifies how the control searches for the objects while user types characters.
item	Returns a specific Item object giving its identifier or caption.
Items	Retrieves the control's Items collection.
LocalAppearance	Retrieves or sets the local popup's appearance.
MenuAppearance	Retrieves or sets a value that indicates the menu's appearance.
Notifier	Retrieves or sets the handle of the window that receives notifications/WM_COMMAND messages.

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.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
Select	Displays the shortcut menu at the specified location and tracks the selection of items on the menu.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowCheckedAsSelected	Specifies whether the checked items shows as selected.
ShowCheckedAsSelectedTransparency	Specifies the transparency (percent) to show the checked items when selected.
ShowPopupArrow	Indicates the type of the arrow to be shown when the item displays the sub-menu.
ShowPopupEffect	Specifies the effect to show the popup menu when clicking an item, such as scrolling, lighting up, and so on.
Template	Specifies the control's template.
TemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.
ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipFont	Retrieves or sets the tooltip's font.
ToolTipPopDelay	Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
ToString	Loads or saves the Items collection using string representation (shortcut of Items.ToString property).
UseVisualTheme	Specifies whether the control uses the current visual theme to display certain UI parts.

Version	Retrieves the control's version.
Visibility	Specify the popup's visibility in percents: 90% is barely visible, and 10% is nearly opaque.
VisualAppearance	Retrieves the control's appearance.

property ExContextMenu.AllowToggleRadio as Boolean

Allows or prevents toggling the radio state.

Type	Description
Boolean	A Boolean expression that specifies whether the radio-buttons allow toggling its value.

By default, the AllowToggleRadio property is False. The AllowToggleRadio property on True, allows a radio button to set on zero (unchecked), if the user clicks twice the radio button. Usually, clicking a radio-button makes the previously checked radio-button in the same group, to be un-checked, and the newly clicked item to be checked. Now, if the AllowToggleRadio property is True, clicking again the radio-button, allows the radio-button to be un-checked, so allows a radio group to have no radio button checked. The control fires the CheckItem event once a radio-button is clicked. The Radio property specifies whether the item displays a radio-button. The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked. The Checked property specifies whether the item is checked or un-checked. The GetRadio method gets a safe array with the radio-items being checked within a radio group. Use the Background(exRadioButtonState0)/Background(exRadioButtonState1) property to specify the visual appearance of the radio-buttons in the control. Use the UseVisualTheme property to specify whether the visual appearance for the radio-buttons to be as indicated by the current XP theme.

property ExContextMenu.AllowToolTip as Boolean

Allows or prevents showing the item's tooltip.

Type	Description
Boolean	A Boolean expression that specifies whether the control displays the item's tooltip when the cursor hovers the item.

By default, the AllowToolTip property is True. Use the AllowToolTip property on False, to prevent shown the item's tooltip when the cursor hovers the item. The Tooltip property assigns a HTML tooltip to an item, that's displayed only when the cursor hovers the item. The TooltipTitle property specifies the title for the item's tooltip. The TooltipDelay property specifies the time until the tooltip is shown. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipFont property specifies 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.

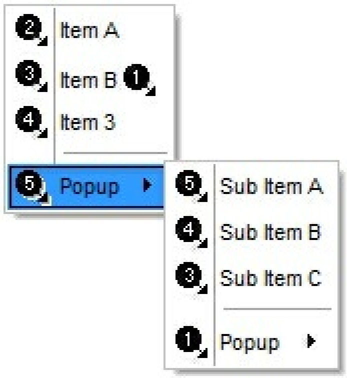
property ExContextMenu.Appearance as MenuBorderEnum

Retrieves or sets the control's appearance.

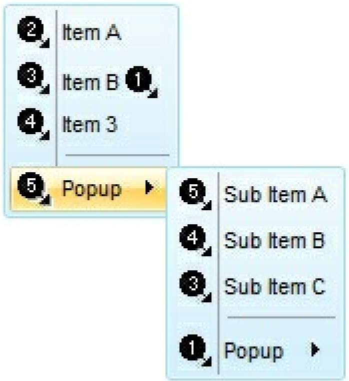
Type	Description
MenuBorderEnum	A MenuBorderEnum expression that specifies the menu's frame appearance, or a color expression whose last 7 bits in the high significant byte of the value indicates the index of the skin in the Appearance collection, being displayed as control's borders. For instance, if 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 normal.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file***

By default, the Appearance property is ShadowBorder. The Appearance property specifies the menu's frame appearance. The SelBackColor property specifies the visual appearance of the item being selected / highlighted. The BackColor property specifies the control's background color. The Background property specifies the visual appearance for different parts of the control, including the radio-buttons, check-boxes or separator items. The LocalAppearance property specifies the visual appearance of the local popup. The PopupAppearance specifies a different visual appearance for the current submenu. When using EBN appearance, using the PopupAppearance, LocalAppearance or Appearance, the distance between margins/borders and items client area is indicated by the client object of the skin/ebn object.

The following screen shot shows the control's frame as displayed by default:



The following screen shot shows the control's frame using a different EBN file:


2
Item A
3
Item B
1
4
Item 3
5
Popup
5
Sub Item A
4
Sub Item B
3
Sub Item C
1
Popup

method ExContextMenu.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 ContextMenu1_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.

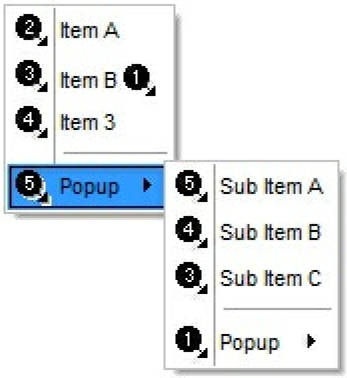
property ExContextMenu.BackColor as Color

Specifies the control's background color.

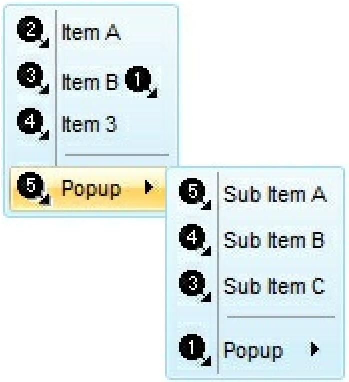
Type	Description
Color	A Color expression that indicates the control's background color.

The BackColor property specifies the control's background color. Use the FlatBackColor property to specify the background color of the left side of the control. The ForeColor property specifies the control's foreground color. The SelBackColor property specifies the visual appearance of the item being selected / highlighted. The SelForeColor property specifies the foreground color of the item being selected / highlighted. The Background property specifies the visual appearance for different parts of the control. The Appearance property specifies the menu's frame appearance. The BackColor property of the Item object specifies a different background color / visual appearance for the entire item.

The following screen shot shows the control's frame as displayed by default:



The following screen shot shows the control's frame using a different EBN file:



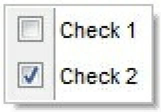
property ExContextMenu.Background(Part as BackgroundPartEnum) as Color

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

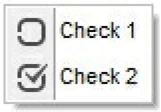
Type	Description
Part as BackgroundPartEnum	A BackgroundPartEnum expression that indicates the part to be changed
Color	A Color expression that indicates the background color for a specified part. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being 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.

Use the Background property to specify a different visual appearance for parts of the control, such as tooltip, check or radio buttons.

The following screen shot shows the check-boxes, as they are shown by default:



The following screen shot shows the check-boxes, as once a new visual appearance is applied:



The following samples show how you can change the visual appearance of the check-boxes:

VB6,VBA (MS Access, Excell...),VB.NET for /COM

```
With CreateObject("Exontrol.ContextMenu")
    With .VisualAppearance
        .Add
1,"gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIKEE
 & _
```

```
"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 & _

"nlapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0
 & _

"cBpbT7CS40JhNEbvJqcZxpT56lwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta
 & _

"qFaa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1
 & _

"geYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GIuIpeB6AoMliBgbD2XJxnYJhhEyC
 & _

"sYRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmIYZ0nCE5LGkRBbhSmJWEYXv
 & _
       "Xg8hmXBThYahCFAECAg=="
       .Add
2,"gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIKEEws
 & _

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 & _

"nlapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0
 & _

"cBpbT7CS40JhNEbvJqcZxpT56lwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSt
 & _

"xAOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+
 & _

"l0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dIHleRAlAEgIA="
    End With
    .Background(70) = &H2000000
    .Background(71) = &H1000000
    .SelBackColor = RGB(240,240,240)
    .SelForeColor = RGB(0,0,0)
    With .Items
        .Add("Check 1",0).Check = 1
        With .Add("Check 2",0)
            .Check = 2
            .Checked = True
        End With
    End With
    .Select
End With
```

VB.NET

```
' Add 'exontrol.excontextmenu.dll' reference to your project.
With New exontrol.EXCONTEXTMENULib.excontextmenu()
    With .VisualAppearance

.Add(1,"gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAw
 & _

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA
 & _

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB
 & _

"cBpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSt
 & _

"qFaa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1
 & _

"geYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GIuIpeB6AoMliBgbD2XJxnYJhhEyC
 & _
"sYRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmIYZ0nCE5LGkRBbhSmJWEYX
 & _
       "Xg8hmXBThYahCFAECAg==")

.Add(2,"gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjI
 & _

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA
 & _

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB
 & _

"cBpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXS
 & _

"xAOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+
 & _

"I0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dIHleRAlAEgIA=")
   End With
   .set_Background32(70,&H2000000)
   .set_Background32(71,&H1000000)
   .SelBackColor = Color.FromArgb(240,240,240)
   .SelForeColor = Color.FromArgb(0,0,0)
   With .Items
      .Add("Check 1",0).Check = True
      With .Add("Check 2",0)
         .Check = True
         .Checked = True
      End With
   End With
   .Select()
End With
```

C++

```
/*
   Includes the definition for CreateObject function like follows:
   #include <comdef.h>
   IUnknownPtr CreateObject( BSTR Object )
   {
      IUnknownPtr spResult;
      spResult.CreateInstance( Object );
      return spResult;
   };
*/
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXCONTEXTMENULib' for the library: 'ExContextMenu
1.0 Type Library'
   #import <ExContextMenu.dll>
   using namespace EXCONTEXTMENULib;
*/
EXCONTEXTMENULib::IExContextMenuPtr var_ExContextMenu =
::CreateObject(L"Exontrol.ContextMenu");
   EXCONTEXTMENULib::IAppearancePtr var_Appearance = var_ExContextMenu-
>GetVisualAppearance();
      var_Appearance-
>Add(1,_bstr_t("gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoDDcNgw
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta
 +

"qFaa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1
 +
"geYBWCklJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GluIpeB6AoMliBgbD2XJxnYJhhEyC
 +

"sYRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmlYZ0nCE5LGkRBbhSmJWEYX
 +
        "Xg8hmXBThYahCFAECAg==");
        var_Appearance-
>Add(2,_bstr_t("gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDDcNgwC
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXS
 +

"xAOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+
 +

"l0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dIHleRAlAEgIA=");
        var_ExContextMenu-
>PutBackground(EXCONTEXTMENULib::exCheckBoxState0,0x2000000);
        var_ExContextMenu-
>PutBackground(EXCONTEXTMENULib::exCheckBoxState1,0x1000000);
        var_ExContextMenu->PutSelBackColor(RGB(240,240,240));
        var_ExContextMenu->PutSelForeColor(RGB(0,0,0));
        EXCONTEXTMENULib::IItemsPtr var_Items = var_ExContextMenu->GetItems();
           var_Items->Add(L"Check 1",long(0),vtMissing)->PutCheck(VARIANT_TRUE);
           EXCONTEXTMENULib::IItemPtr var_item = var_Items->Add(L"Check
2",long(0),vtMissing);
              var_item->PutCheck(VARIANT_TRUE);
              var_item->PutChecked(VARIANT_TRUE);
        var_ExContextMenu->Select(vtMissing,vtMissing,vtMissing);
```

C++ Builder

```
/*
   Select the Component\Import Component...\Import a Type Library,
   to import the following Type Library:
      ExContextMenu 1.0 Type Library
   TypeLib: e:\Exontrol\ExContextMenu\project\Site\ExContextMenu.dll
   to define the namespace: Excontextmenulib_tlb
*/
//#include "EXCONTEXTMENULIB_TLB.h"
Excontextmenulib_tlb::IExContextMenuPtr var_ExContextMenu =
Variant::CreateObject(L"Exontrol.ContextMenu");
   Excontextmenulib_tlb::IAppearancePtr var_Appearance = var_ExContextMenu-
>VisualAppearance;
      var_Appearance-
>Add(1,TVariant(String("gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAo
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta
 +

"qFaa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1
 +

"geYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GluIpeB6AoMIiBgbD2XJxnYJhhEyC
 +

"sYRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmIYZ0nCE5LGkRBbhSmJWEYX
 +
      "Xg8hmXBThYahCFAECAg=="));
      var_Appearance-
```

```
>Add(2,TVariant(String("gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoD
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSt
 +

"xAOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+
 +

"l0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dIHleRAlAEgIA="));
	var_ExContextMenu-
>set_Background(Excontextmenulib_tlb::BackgroundPartEnum::exCheckBoxState0,0x20

	var_ExContextMenu-
>set_Background(Excontextmenulib_tlb::BackgroundPartEnum::exCheckBoxState1,0x10

	var_ExContextMenu->SelBackColor = RGB(240,240,240);
	var_ExContextMenu->SelForeColor = RGB(0,0,0);
	Excontextmenulib_tlb::IItemsPtr var_Items = var_ExContextMenu->Items;
		var_Items->Add(L"Check 1",TVariant(0),TNoParam())->Check = true;
		Excontextmenulib_tlb::IItemPtr var_item = var_Items->Add(L"Check
2",TVariant(0),TNoParam());
			var_item->Check = true;
			var_item->Checked = true;
	var_ExContextMenu->Select(TNoParam(),TNoParam(),TNoParam());
```

C#

```
// Add 'exontrol.excontextmenu.dll' reference to your project.
exontrol.EXCONTEXTMENULib.excontextmenu var_ExContextMenu = new
exontrol.EXCONTEXTMENULib.excontextmenu();
```

```
    exontrol.EXCONTEXTMENULib.Appearance var_Appearance =
var_ExContextMenu.VisualAppearance;

var_Appearance.Add(1,"gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoD
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSt
 +

"qFaa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1
 +

"geYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GluIpeB6AoMIiBgbD2XJxnYJhhEyC
 +

"sYRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmIYZ0nCE5LGkRBbhSmJWEYX
 +
       "Xg8hmXBThYahCFAECAg==");

var_Appearance.Add(2,"gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDD
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSt
 +
"xAOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+
 +

"l0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dlHleRAlAEgIA=");

var_ExContextMenu.set_Background32(exontrol.EXCONTEXTMENULib.BackgroundPart


var_ExContextMenu.set_Background32(exontrol.EXCONTEXTMENULib.BackgroundPart

        var_ExContextMenu.SelBackColor = Color.FromArgb(240,240,240);
        var_ExContextMenu.SelForeColor = Color.FromArgb(0,0,0);
        exontrol.EXCONTEXTMENULib.Items var_Items = var_ExContextMenu.Items;
            var_Items.Add("Check 1",0,null).Check = true;
            exontrol.EXCONTEXTMENULib.item var_item = var_Items.Add("Check
2",0,null);
                var_item.Check = true;
                var_item.Checked = true;
        var_ExContextMenu.Select(null,null,null);
```

C# for /COM

```
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
EXCONTEXTMENULib.ExContextMenu var_ExContextMenu = new
EXCONTEXTMENULib.ExContextMenu();
    EXCONTEXTMENULib.Appearance var_Appearance =
var_ExContextMenu.VisualAppearance;

var_Appearance.Add(1,"gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoD
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta
 +

"qFaa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1
 +

"geYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GIuIpeB6AoMliBgbD2XJxnYJhhEyC
 +

"sYRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmIYZ0nCE5LGkRBbhSmJWEYX
 +
       "Xg8hmXBThYahCFAECAg==");

var_Appearance.Add(2,"gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDD
 +

"iEZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

"nIapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0
 +

"cBpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSt
 +

"xAOap0nmXYIE8Y4zkabZAkofgsCuZ5LI6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+
 +

"l0FRcgOApZggNgOgKSA2HGERjIsEZaBaA4ZGgWB2GwW4oE2dIHIeRAIAEgIA=");

var_ExContextMenu.set_Background(EXCONTEXTMENULib.BackgroundPartEnum.exChe


var_ExContextMenu.set_Background(EXCONTEXTMENULib.BackgroundPartEnum.exChe

       var_ExContextMenu.SelBackColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(240,240,240));
```

```
        var_ExContextMenu.SelForeColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0));
        EXCONTEXTMENULib.Items var_Items = var_ExContextMenu.Items;
            var_Items.Add("Check 1",0,null).Check = true;
            EXCONTEXTMENULib.item var_item = var_Items.Add("Check 2",0,null);
                var_item.Check = true;
                var_item.Checked = true;
        var_ExContextMenu.Select(null,null,null);
```

X++ (Dynamics Ax 2009)

```
COM com_Appearance,com_ExContextMenu,com_Items,com_item;
anytype var_Appearance,var_ExContextMenu,var_Items,var_item;
str var_s,var_s1;
;
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
var_ExContextMenu = COM::createFromObject(new
EXCONTEXTMENULib.excontextmenu()); com_ExContextMenu = var_ExContextMenu;
    var_Appearance = com_ExContextMenu.VisualAppearance(); com_Appearance =
var_Appearance;
        var_s =
"gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIKEEws

        var_s = var_s +
"EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7

        var_s = var_s +
"IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0C

        var_s = var_s +
"BpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta

        var_s = var_s +
"Faa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1x

        var_s = var_s +
"eYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GluIpeB6AoMliBgbD2XJxnYJhhEyOIl
```

```
	var_s = var_s +
"YRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmlYZ0nCE5LGkRBbhSmJWEYXw

	var_s = var_s + "g8hmXBThYahCFAECAg==";
	com_Appearance.Add(1,COMVariant::createFromStr(var_s));
	var_s1 =
"gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIKEEwsA

	var_s1 = var_s1 +
"EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7

	var_s1 = var_s1 +
"IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0C

	var_s1 = var_s1 +
"BpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta

	var_s1 = var_s1 +
"AOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+F

	var_s1 = var_s1 +
"0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dIHleRAlAEgIA=";
	com_Appearance.Add(2,COMVariant::createFromStr(var_s1));
   com_ExContextMenu.Background(70/*exCheckBoxState0*/,0x2000000);
   com_ExContextMenu.Background(71/*exCheckBoxState1*/,0x1000000);
   com_ExContextMenu.SelBackColor(WinApi::RGB2int(240,240,240));
   com_ExContextMenu.SelForeColor(WinApi::RGB2int(0,0,0));
   var_Items = com_ExContextMenu.Items(); com_Items = var_Items;
	var_item = COM::createFromObject(com_Items.Add("Check
1",COMVariant::createFromInt(0))); com_item = var_item;
	com_item.Check(1);
	var_item = com_Items.Add("Check 2",COMVariant::createFromInt(0)); com_item =
var_item;
	   com_item.Check(2);
	   com_item.Checked(true);
   com_ExContextMenu.Select();
```

Delphi 8 (.NET only)

```
with (ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMenu'))
as EXCONTEXTMENULib.ExContextMenu) do
begin
  with VisualAppearance do
  begin

Add(1,'gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwj
 +

'EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

'IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0C
 +

'BpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXStap
 +

'Faa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1x
 +

'eYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GluIpeB6AoMliBgbD2XJxnYJhhEyOIl
 +

'YRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmIYZ0nCE5LGkRBbhSmJWEYXw
 +
    'g8hmXBThYahCFAECAg==');

Add(2,'gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIK
 +

'EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +

'IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0C
```

```
 +

'BpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta
 +

'AOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+P
 +

'0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dIHleRAlAEgIA=');
  end;
  Background[70] := $2000000;
  Background[71] := $1000000;
  SelBackColor := $f0f0f0;
  SelForeColor := $0;
  with Items do
  begin
    Add('Check 1',TObject(0),Nil).Check := True;
    with Add('Check 2',TObject(0),Nil) do
    begin
      Check := True;
      Checked := True;
    end;
  end;
  Select(Nil,Nil,Nil);
end;
```

Delphi (standard)

```
with
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMen
 as EXCONTEXTMENULib_TLB.ExContextMenu) do
begin
  with VisualAppearance do
  begin

Add(1,'gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwj
 +
'EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +
'IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0C
 +
'BpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta
 +
'Faa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1x
 +
'eYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GluIpeB6AoMliBgbD2XJxnYJhhEyOIl
 +
'YRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmIYZ0nCE5LGkRBbhSmJWEYXw
 +
      'g8hmXBThYahCFAECAg==');

Add(2,'gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIK
 +
'EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7
 +
'IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0C
 +
'BpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta
 +
'AOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+P
 +
'0FRcgOApZggNgOgKSA2HGERjIsEZaBaA4ZGgWB2GwW4oE2dIHleRAIAEgIA=');
   end;
```

```
    Background[70] := $2000000;
    Background[71] := $1000000;
    SelBackColor := $f0f0f0;
    SelForeColor := $0;
    with Items do
    begin
        Add('Check 1',OleVariant(0),Null).Check := True;
        with Add('Check 2',OleVariant(0),Null) do
        begin
            Check := True;
            Checked := True;
        end;
    end;
    Select(Null,Null,Null);
end;
```

VFP

```
with CreateObject("Exontrol.ContextMenu")
    with .VisualAppearance
        var_s =
"gBFLBCJwBAEHhEJAEGg4BVMMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIKEEws

        var_s = var_s +
"EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7

        var_s = var_s +
"IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0C

        var_s = var_s +
"BpbT7CS40JhNEbvJqcZxpT56IwhPZdQrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXStap

        var_s = var_s +
"Faa4xHsOZMi8P4jHwbZ4DQRZOj+EIsGKc46n0NYumUYgHmyPg5n4JhPh+CQVnacp1x

        var_s = var_s +
"eYBWCkIJDE4Dh8kYRw8FOBJYFOZgWFaCYIGSd4GluIpeB6AoMliBgbD2XJxnYJhhEyOIl
```

```
    var_s = var_s +
"YRGAiZY8gqWJznYPhvB0URoH6EJaiYRRXCCZIGGIShhmlYZ0nCE5LGkRBbhSmJWEYXw

    var_s = var_s + "g8hmXBThYahCFAECAg=="
    .Add(1,var_s)
    var_s1 =
"gBFLBCJwBAEHhEJAEGg4BJkMQAAYAQGKIYBkAKBQAGaAoDDcNgwQwAAwjIKEEwsA

    var_s1 = var_s1 +
"EZRQiiCYsS5GQBSFDcOwHGyQYDkCQpAAWL4tCyNc7QHKFAxnAgaaLiIZZUh+TQAA7

    var_s1 = var_s1 +
"IapZDKGKQAKhQgiNqqGg2QiKFRXHSgMQuaClKbiaqqaTGfh7YAUGBEbgmC4NQjB0

    var_s1 = var_s1 +
"BpbT7CS40JhNEbvJqcZxpT56IwmRC5QrPVZrKCcLwVSa3ahuO5bOxOC4XWaBcRwXSta

    var_s1 = var_s1 +
"AOap0nmXYIE8Y4zkabZAkofgsCuZ5Ll6VB5F8OBfBET4WH2d5hFkfwvD4c5kkuQp7k+

    var_s1 = var_s1 +
"0FRcgOApZggNgOgKSA2HGERjlsEZaBaA4ZGgWB2GwW4oE2dIHleRAIAEgIA="
    .Add(2,var_s1)
  endwith
  .Background(70) = 0x2000000
  .Background(71) = 0x1000000
  .SelBackColor = RGB(240,240,240)
  .SelForeColor = RGB(0,0,0)
  with .Items
    .Add("Check 1",0).Check = 1
    with .Add("Check 2",0)
      .Check = 2
      .Checked = .T.
    endwith
  endwith
  .Select()
```

endwith

property ExContextMenu.CloseOnClick as CloseOnClickEnum

Gets or sets a value that specifies whether the context menu is closing.

Type	Description
CloseOnClickEnum	A CloseOnClickEnum expression that specifies how the user can close the context menu.

By default, the CloseOnClick property is exCloseOnNonClickable, and it means that the context menu is closed once the user clicks a regular item with no sub menus, check or radio buttons. Use te CloseOnClick property to specify how the user can close the context menu. The Select method returns the identifier of the last clicked item. Use the Item's CloseOnClick property to specify a different way to close the menu when user clicks a specified item.

property ExContextMenu.Cursor as Variant

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

Type	Description
Variant	A String expression that defines the cursor to be shown when the cursor hovers the menu. The Valid values are listed bellow. Also the Cursor property could point to a cursor file to be loaded and shown while the cursor hovers the context menu.

By default, the Cursor property is "exDefault". Use the Cursor property to specify a different cursor when it hovers the menu control. Use the Cursor property of the Item object to specify a different cursor when it hovers the item only.

The supported values are:

- "exDefault", Standard arrow
- "exArrow", Standard arrow
- "exCross", Crosshair
- "exIBeam", I-beam
- "exIcon", Reserved
- "exSize", Reserved, use the "exSizeAll"
- "exSizeNESW", Double-pointed arrow pointing northeast and southwest
- "exSizeNS", Double-pointed arrow pointing north and south
- "exSizeNWSE", Double-pointed arrow pointing northwest and southeast
- "exSizeWE", Double-pointed arrow pointing west and east
- "exUpArrow", Vertical arrow
- "exHourglass", Hourglass
- "exNoDrop", Slashed circle
- "exArrowHourglass"
- "exHelp", Arrow and question mark
- "exSizeAll", Four-pointed arrow pointing north, south, east, and west
- "exHand", Hand

Any other value indicates the path to a cursor file to be displayed when the pointer hovers the menu control.

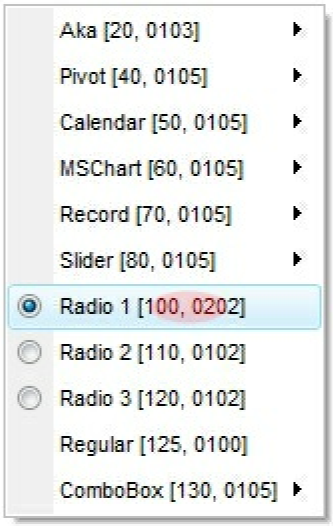
property ExContextMenu.Debug as Boolean

Retrieves or sets a value that indicating whether the item's identifier is visible.

Type	Description
Boolean	A Boolean expression that specifies whether the identifiers of the items

By default, the Debug property is False. Use the Debug property to display the identifiers for all visible items, for debugging purposes. The First number in the [] parenthesis indicates the item's ID property.

The following screen shot shows the control with the Debug property on True:



property ExContextMenu.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 ExContextMenu.ExecuteTemplate (Template as String)

Executes a template and returns the result.

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

Use the ExecuteTemplate property to returns the result of executing a template file. Use the Template property to execute a template without returning any result. Use the ExecuteTemplate property to execute code by passing instructions as a string (template string).

For instance, the following sample adds a few items, displays the context menu and returns the selected identifier :

```
Set n = New EXCONTEXTMENULib.ExContextMenu
Debug.Print (n.ExecuteTemplate("Items.ToString = `Item A[id=1001],Item B,Item C,Item D`;Select()"))
```

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.

The Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as a progID
<text> any string of characters

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 ExContextMenu.FlatBackColor as Color

Specifies the color to left part of the menu.

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

Use the FlatBackColor property to specify the background color of the left side of the control. This property has effect while the control's MenuAppearance property is exMenuFlat. The BackColor property specifies the control's background color. The ForeColor property specifies the control's foreground color. The SelBackColor property specifies the visual appearance of the item being selected / highlighted. The SelForeColor property specifies the foreground color of the item being selected / highlighted. The Background property specifies the visual appearance for different parts of the control. The Appearance property specifies the menu's frame appearance. The Background(exMenuFlatLineColor) property indicates the color of line that divides the left to right side of the menu, when the MenuAppearance property is exMenuFlat.

property ExContextMenu.FlatImageWidth as Long

Specifies the width of the column to display the icons/images when the control's MenuAppearance is exMenuFlat.

Type	Description
Long	A Long expression that specifies the width of the column that displays icons/images/check or radio buttons, when the control's MenuAppearance is exMenuFlat.

By default, the FlatImageWidth property is 16 pixels wide. Use the FlatImageWidth property to specify the width of the column that displays icons/images/check or radio buttons. The Image / HTMLImage property assigns an icon / picture to the item. The <img> tag can be used in the Caption property of the Item object to display an Icon or a custom-size picture.

property ExContextMenu.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object to be used to shown the control items.

Use the Font property to specify a different font to show the items in the context menu. The Font's height controls the height of the items in the control. You can use the <font> HTML tag to specify a different font for a specified item in the Caption property. The ForeColor property of the control specifies the foreground color of items in the control.

property ExContextMenu.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. The BackColor property specifies the control's background color. The SelBackColor property specifies the visual appearance of the item being selected / highlighted. The SelForeColor property specifies the foreground color of the item being selected / highlighted. The Background property specifies the visual appearance for different parts of the control. The Appearance property specifies the menu's frame appearance. You can use the <fgcolor> HTML tag to specify a different foreground colot for a specified item in the Caption property. The ForeColor property of the Item object specifies a different foreground color for the entire item.

property ExContextMenu.Get (Criteria as MenuItemTypeEnum) as Variant

Retrieves an array of Item objects that meet the criteria.

Type	Description
Criteria as MenuItemTypeEnum	A MenuItemTypeEnum expression that type of items to be retrieved.
Variant	A Safe-Array of Item objects being returned.

The Get method can be used to get a collection / safe array of Item objects with a specified characteristics. For instance, you can collect the items of Edit type, or items that displays an icon using the Image property. The GetChecked property gets a collection of checked items. The GetUnchecked property gets a collection of checked items. The GetRadio method gets a safe array with the radio-items being checked within a radio group. For instance, the GetChecked property is equivalent with the Get(exCheckBoxMenuItem + exCheckedMenuItem), or in other words all items with the Check and Checked properties on True. The result of the Get method indicates a Safe-Array of Item objects, which means that you can use the **for each** statement to enumerate the elements in the collection. The ItemType property is a read-only property that gets the type of the item.

property ExContextMenu.GetChecked as Variant

Retrieves an array of Item objects, that displays a check box which is checked.

Type	Description
Variant	A Safe-Array of Item objects that indicates the checked items in the control. The collection does include only items with the Check property set on True.

The GetChecked property gets a collection of checked items. The GetUnchecked property gets a collection of checked items. The GetRadio method gets a safe array with the radio-items being checked within a radio group. The Check property indicates whether the current item displays a check box. The Checked property specifies whether the item is checked or un-checked. The Radio property specifies whether the item displays a radio-button. The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked inside.

The following VB sample displays the caption of the items being checked in the control:

```
Dim c As Variant
For Each c In contextMenu.GetChecked
   Debug.Print vbTab & c.Caption
Next
```

The following VB/NET sample displays the caption of the items being checked in the control:

```
Dim c As Object
For Each c In Excontextmenu1.GetChecked
   Debug.Print(vbTab & c.Caption)
Next
```

The following C# sample displays the caption of the items being checked in the control:

```
foreach (exontrol.EXCONTEXTMENULib.Item i in excontextmenu1.GetChecked)
   System.Diagnostics.Debug.Print("\t" + i.Caption);
```

property ExContextMenu.GetRadio ([RadioGroup as Variant]) as Variant

Retrieves an array of Item objects of radio type in the same group, that are checked.

Type	Description
RadioGroup as Variant	A Long expression that specifies the radio-group being queried, or zero if you were not used any RadioGroup call.
Variant	A Safe-Array of Item objects that indicates the radio-checked items in the control. The collection does include only items with the Radio property set on True. The collection may contains zero or one element indicating the radio-item being checked in the specified radio group.

The GetRadio method gets a safe array with the radio-items being checked within a radio group. The GetChecked property gets a collection of checked items. The GetUnchecked property gets a collection of checked items. The Check property indicates whether the current item displays a check box. The Checked property specifies whether the item is checked or un-checked. The Radio property specifies whether the item displays a radio-button. The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked inside.

The following VB sample displays the caption of radio-item being checked (single radio-group, or the RadioGroup property has not been used to create ore groups) :

```
Dim c As Variant
For Each c In contextMenu.GetRadio
   Debug.Print vbTab & c.Caption
Next
```

The following VB sample displays the caption of radio-item being checked in the radio-group with the identifier 100:

```
Dim c As Variant
For Each c In contextMenu.GetRadio(100)
   Debug.Print vbTab & c.Caption
Next
```

The following VB/NET sample displays the caption of radio-item being checked (single radio-group, or the RadioGroup property has not been used to create ore groups) :

```
Dim c As Variant
```

```
For Each c In contextMenu.GetRadio
   Debug.Print vbTab & c.Caption
Next
```

The following VB/NET sample displays the caption of radio-item being checked in the radio-group with the identifier 100:

```
Dim c As Variant
For Each c In contextMenu.get_GetRadio(100)
   Debug.Print vbTab & c.Caption
Next
```

The following C# sample displays the caption of radio-item being checked (single radio-group, or the RadioGroup property has not been used to create ore groups) :

```
foreach (exontrol.EXCONTEXTMENULib.Item i in excontextmenu1.GetRadio)
   System.Diagnostics.Debug.Print("\t" + i.Caption);
```

The following C# sample displays the caption of radio-item being checked in the radio-group with the identifier 100:

```
foreach (exontrol.EXCONTEXTMENULib.Item i in excontextmenu1.get_GetRadio(100))
   System.Diagnostics.Debug.Print("\t" + i.Caption);
```

property ExContextMenu.GetUnchecked as Variant

Retrieves an array of Item objects, that displays a check box which is unchecked.

Type	Description
Variant	A Safe-Array of Item objects that indicates the checked items in the control. The collection does include only items with the Check property set on True.

The GetUnchecked property gets a collection of checked items. The GetChecked property gets a collection of checked items. The GetRadio method gets a safe array with the radio-items being checked within a radio group. The Check property indicates whether the current item displays a check box. The Checked property specifies whether the item is checked or un-checked. The Radio property specifies whether the item displays a radio-button. The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked inside.

The following VB sample displays the caption of the items being un-checked in the control:

```
Dim c As Variant
For Each c In contextMenu.GetUnchecked
   Debug.Print vbTab & c.Caption
Next
```

The following VB/NET sample displays the caption of the items being un-checked in the control:

```
Dim c As Object
For Each c In Excontextmenu1.GetUnchecked
   Debug.Print(vbTab & c.Caption)
Next
```

The following C# sample displays the caption of the items being un-checked in the control:

```
foreach (exontrol.EXCONTEXTMENULib.Item i in excontextmenu1.GetUnchecked)
   System.Diagnostics.Debug.Print("\t" + i.Caption);
```

property ExContextMenu.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

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

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> tags. By default, the HTMLPicture collection is empty. Use the HTMLPicture property to add new pictures to be used in HTML captions. For instance, the HTMLPicture("pic1") = "c:\winnt\zapotec.bmp", loads the zapotec picture and associates the pic1 key to it. Any "<img>pic1</img>" sequence in HTML captions, displays the pic1 picture. On return, the HTMLPicture property retrieves a Picture object (this implements the IPictureDisp interface). The <img> tag can be used in the Caption property of the Item object. Use the HTMLImage property to assign a BMP, JPG, GIF or PNG file to left side of the caption, the same way as you will do with the Image property. Use the FlatImageWidth property to specify the width of the column that displays icons/images/check or radio buttons.

method ExContextMenu.Images (Handle as Variant)

Sets the control's image list at runtime.

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

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

The user can add images at design time, by drag and drop files to combo's image holder. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The <img> tag can be used in the Caption property of the Item object. Also, the Image property assign an icon to the specified item.

property ExContextMenu.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 ExContextMenu.IncrementalSearch as IncrementalSearchEnum

Specifies how the control searches for the objects while user types characters.

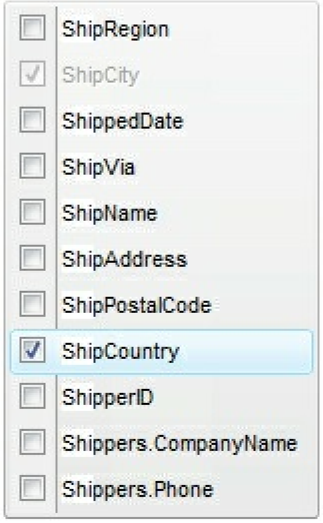
Type	Description
IncrementalSearchEnum	An IncrementalSearchEnum expression that specifies the type of incremental searching the control performs once the user types characters on the context menu.

"In computing, incremental search, incremental find or real-time suggestions is a user interface interaction method to progressively search for and filter through text. As the user types text, one or more possible matches for the text are found and immediately presented to the user. " By default, the IncrementalSearch property is exISearchStartWith + exISearchFilterFor, in other words, the control filter for items that match the typing characters. Use the IncrementalSearch property on exNoIncrementalSearch to disable/prevent the incremental searching in your context menu. While the incremental search is on, the F3 or Shift + F3, finds the next occurrence or previously occurrence. The Back key deletes the last character of the incremental search string, while the Ctrl + Back key removes the entire incremental search string. If the IncrementalSearch property is exNoIncrementalSearch, you can use the item's Shortcut property to define the key combination that the user can press to select the item quickly.

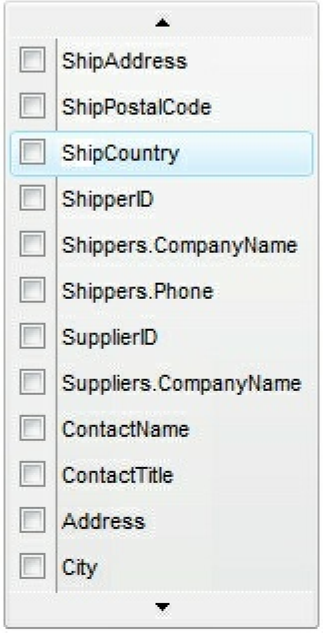
You can use the IncrementalSearch property:

- to highlight the items that match the typing characters
- to display just the items that match the typing characters

The following screen shot shows the control with IncrementalSearch property on exISearchStartWith + exISearchFilterFor, while the user types "**shi**":



he following screen shot shows the control with IncrementalSearch property on exISearchStartWith, while the user types "**shi**":


▲
☐ ShipAddress
☐ ShipPostalCode
☐ ShipCountry
☐ ShipperID
☐ Shippers.CompanyName
☐ Shippers.Phone
☐ SupplierID
☐ Suppliers.CompanyName
☐ ContactName
☐ ContactTitle
☐ Address
☐ City
▼

property ExContextMenu.item (ID as Variant) as Item

Returns a specific Item object giving its identifier or caption.

Type	Description
ID as Variant	A Long expression that specifies the identifier of the item being requested or a String expression that specifies the caption of the item being requested.
Item	An Item object with associated identifier.

The Item property searches recursively the item with giving identifier/caption. The ID property of the Item object specifies the identifier of the item. The Caption property of the Item object specifies the caption of the item. The Item property gets the first Item object being found, if multiple objects with the same identifier are found, or Nothing, if no item with associated identifier is found. The Item property of the Items compared with the Item property of the eXContextMenu is that the first look in the specified Items collection, while the second is looking for all Items in the menu object.

property ExContextMenu.Items as Items

Retrieves the control's Items collection.

Type	Description
Items	An Items object that holds a collection of Item objects.

The Items property gives access to the control's Items collection, so you can add, remove or update the items being shown in the context menu. The Add method adds a new item to the Items collection. The ToString property loads or saves the control items from a string. The Remove method removes a specified item. The Select property shows the context menu, and waits for the user to make the selection.

The following VB sample loads three items (Item A, Item B and Item C) from a string and displays the context menu:

```
Set contextMenu = CreateObject("Exontrol.ContextMenu")
With contextMenu
    .Items.ToString = "Item A,Item B,Item C"
    iSelect = .Select()
    If (iSelect <> 0) Then
        Debug.Print (.Items.Item(iSelect).Caption)
    End If
End With
```

The Item property accesses an Item object giving its identifier or caption.

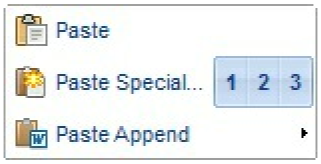
property ExContextMenu.LocalAppearance as MenuBorderEnum

Retrieves or sets the local popup's appearance.

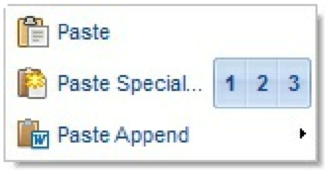
Type	Description
MenuBorderEnum	A MenuBorderEnum expression that specifies the local's frame appearance, or a color expression whose last 7 bits in the high significant byte of the value indicates the index of the skin in the Appearance collection, being displayed as control's borders. For instance, 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 normal.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file***

By default, the LocalAppearance property is -1. The visual appearance of the local popup is specified by the control's Appearance property, while the LocalAppearance property is -1. The ShowLocalPopup property specifies whether the item's popup is shown as local. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus. The PopupAppearance specifies a different visual appearance for the current submenu. When using EBN appearance, using the PopupAppearance, LocalAppearance or Appearance, the distance between margins/borders and items client area is indicated by the client object of the skin/ebn object.

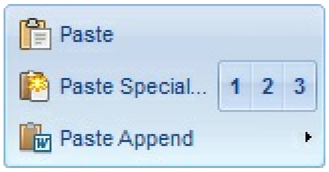
The following screen shot shows the sub-menu with different appearances:



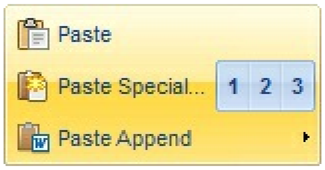
(single appearance)



(shadow appearance)



(ebn appearance)



(ebn appearance)

property ExContextMenu.MenuAppearance as MenuAppearanceEnum

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

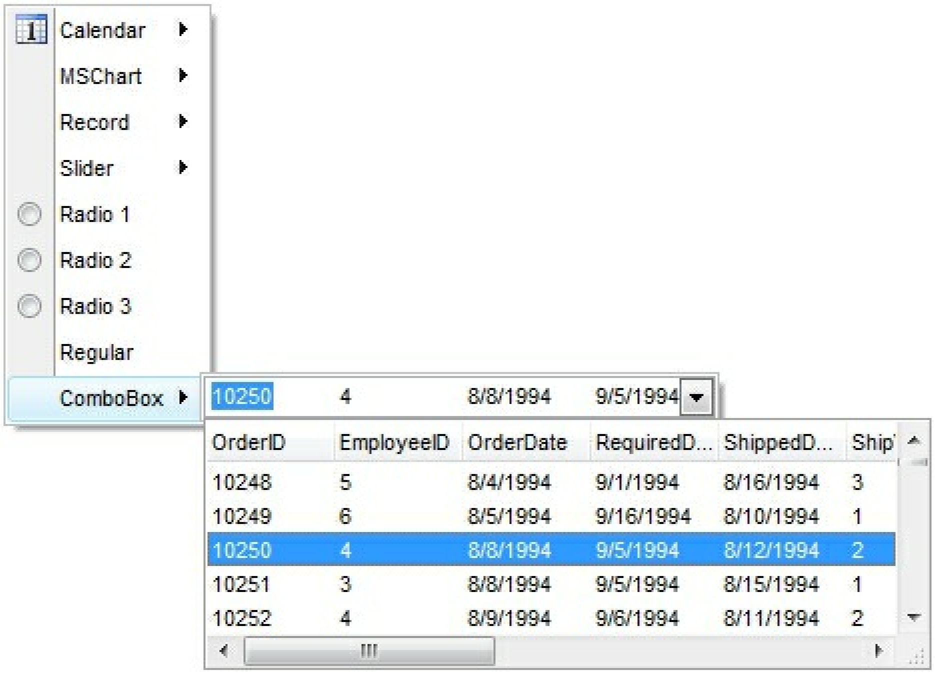
Type	Description
MenuAppearanceEnum	A MenuAppearanceEnum expression that specifies menu's appearance.

By default, the MenuAppearance property is exMenuFlat. The Background(exMenuFlatLineColor) property indicates the color of line that divides the left to right side of the menu, when the MenuAppearance property is exMenuFlat. The FlatBackColor property indicates the color to show the left part of the menu, when the MenuAppearance property is exMenuFlat. The BackColor property specifies the menu's background color. The Background(exMenuButtonItem) property indicates the visual appearance for items in the menu control, when the MenuAppearance property is exMenuButton. The Background(exMenuSeparatorItem) property specifies the visual appearance of the separator items.

The MenuAppearance supports the following values:

- **exMenuNormal**, the BackColor property specifies the menu's background color. The ForeColor property specifies the menu's foreground color.
- **exMenuFlat**, the BackColor property specifies the menu's background color. The ForeColor property specifies the menu's foreground color. The Background(exMenuFlatLineColor) property indicates the color of line that divides the left to right side of the menu. The FlatBackColor property indicates the color to show the left part of the menu.
- **exMenuButton,** the BackColor property specifies the menu's background color. The ForeColor property specifies the menu's foreground color. The Background(exMenuButtonItem) property indicates the visual appearance for items in the menu control.

The following screen shot shows the menu while the MenuAppearance property is exMenuFlat :



10250 4 8/8/1994 9/5/1994

OrderID	EmployeeID	OrderDate	RequiredD...	ShippedD...	Ship'
10248	5	8/4/1994	9/1/1994	8/16/1994	3
10249	6	8/5/1994	9/16/1994	8/10/1994	1
10250	4	8/8/1994	9/5/1994	8/12/1994	2
10251	3	8/8/1994	9/5/1994	8/15/1994	1
10252	4	8/9/1994	9/6/1994	8/11/1994	2

property ExContextMenu.Notifier as Long

Retrieves or sets the handle of the window that receives notifications/WM_COMMAND messages.

Type	Description
Long	A Long expression that specifies the handle of the window that receives the WM_COMMAND when the user selects, check/uncheck, edit an item.

By default, the Notifier property is 0, which indicates that the property has no effect. Set the Notifier property to a window that you want to receive notification of the control through the WM_COMMAND message. For instance, in VFP or C++ it would be easier to handle the events of the control using the WM_COMMAND messages, rather than using sink interfaces.

The wParam parameter of the WM_COMMAND message carries the identifier of the event which occurred like listed bellow:

- **0** (exSelectItem), occurs when the user selects/clicks an item
- **1** (exCheckItem), occurs when the user clicks the item's check box, or check the item's checkbox
- **2** (exUncheckItem), occurs when the user clicks the item's check box, or uncheck the item's checkbox
- **3** (exEditChangeItem), occurs when the content of the item's editor is changed.

The lParam parameter of the WM_COMMAND message carries the identifier of the item who fired the event. You can use the Item property to access the control's item giving its identifier. The ID property specifies the item's identifier.

In VFP, you have to assign the hWnd property of the form to the Notifier property of the control as follows:

```
contextMenu.Notifier = thisform.HWnd
```

while the following code:

```
BINDEVENT( thisform.HWnd, 273, thisform, "oncommand" )
```

adds a handler oncommand for the WM_COMMAND message (273 or 0x111 in hexa, is the identifier of the WM_COMMAND message).

The oncommand may look like:

```
LPARAMETERS hWnd, uMsg, wParam, lParam

?wParam

*CheckItem
IF ( wParam = 1 ) then
    thisform.contextMenu_CheckItem(lParam)
ELSE
    * UncheckItem
    IF ( wParam = 2 ) then
        thisform.contextMenu_UncheckItem(lParam)
    ENDIF
ENDIF
```

property ExContextMenu.Picture as IPictureDisp

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

Type	Description
IPictureDisp	A Picture object that indicates the control's picture.

Reserved. Currently, this property is disabled.

property ExContextMenu.PictureDisplay as PictureDisplayEnum

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

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way how the control's picture is displayed.

Reserved. Currently, this property is disabled.

method ExContextMenu.Refresh ()

Refreshes the control.

Type	Description

Call the Refresh method to update the control's content. For instance, if you are changing the item's Caption, if an OleEvent occurs.

method ExContextMenu.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 Variant expression that specifies the icon to add or insert, as one of the following options: - a long expression that specifies the handle of the icon (HICON) - a string expression that indicates the path to the picture file - a string expression that defines the picture's content encoded as BASE64 strings using the eXImages tool - a Picture reference, which is an object that holds image data. It is often used in controls like PictureBox, Image, or in custom controls (e.g., IPicture, IPictureDisp) If the Icon parameter is 0, it specifies that the icon at the given Index is removed. Furthermore, setting the Index parameter to -1 removes all icons. By default, if the Icon parameter is not specified or is missing, a value of 0 is used.
Index as Variant	A long expression that defines the index of the icon to insert or remove, as follows: - A zero or positive value specifies the index of the icon to insert (when Icon is non-zero) or to remove (when the Icon parameter is zero) - A negative value clears all icons when the Icon parameter is zero By default, if the Index parameter is not specified or is missing, a value of -1 is used.

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

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

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

i = ExContextMenu1.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 = ExContextMenu1.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:

ExContextMenu1.ReplaceIcon 0, i, where i specifies the index of icon removed.

The following VB clears the control's icons collection:

ExContextMenu1.ReplaceIcon 0, -1

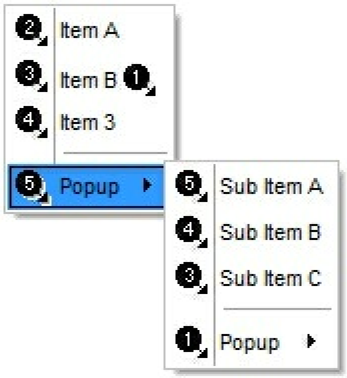
property ExContextMenu.SelBackColor as Color

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

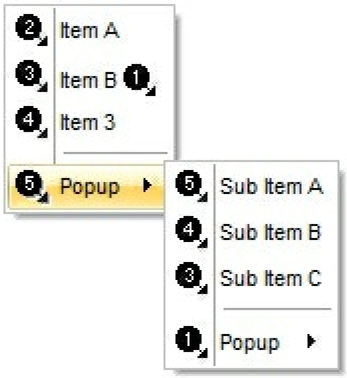
Type	Description
Color	A Color expression that specifies the background color / visual appearance of the selected item. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove 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 SelBackColor property specifies the visual appearance of the item being selected / highlighted. The SelForeColor property specifies the foreground color of the item being selected / highlighted. The ForeColor property specifies the control's foreground color. The BackColor property specifies the control's background color. The Background property specifies the visual appearance for different parts of the control. The Appearance property specifies the menu's frame appearance.

The following screen shot shows the control's selection with the default colors:



The following screen shot shows the control's selection with a different visual appearance:



method ExContextMenu.Select ([Flags as Variant], [X as Variant], [Y as Variant])

Displays the shortcut menu at the specified location and tracks the selection of items on the menu.

Type	Description
Flags as Variant	A Long expression that indicates the alignment of the context menu relative to the giving X and Y parameters. If missing, the 0 is used, so the menu's top-left corner is aligned to X, Y coordinates. The Flags parameter can be a combination of the following values: Use one of the following flags to specify how the function positions the shortcut menu horizontally: • **0** Positions the shortcut menu so that its left side is aligned with the coordinate specified by the x parameter. • **4** Centers the shortcut menu horizontally relative to the coordinate specified by the x parameter. • **8** Positions the shortcut menu so that its right side is aligned with the coordinate specified by the x parameter Use one of the following flags to specify how the function positions the shortcut menu vertically: • **0** Positions the shortcut menu so that its top side is aligned with the coordinate specified by the y parameter • **16** Centers the shortcut menu vertically relative to the coordinate specified by the y parameter • **32** Positions the shortcut menu so that its bottom side is aligned with the coordinate specified by the y parameter For instance, the 4 + 32 indicates that the menu is horizontally centered relative to x, and vertically it under the y coorindate.

X as Variant	If missing or -1, the current cursor position is used, else it should indicate the X position to show the context menu, in screen coordinates.
Y as Variant	If missing or -1, the current cursor position is used, else it should indicate the Y position to show the context menu, in screen coordinates.

Return	Description
Long	A Long expression that specifies the identifier of the Item being clicked. A zero(0) value indicates that the user makes no selection, or no item has been clicked. A value different than zero indicates the item with the specified ID. You can use the Item property of the control to get the associated Item object based on this identifier.

The Select property shows the context menu, and waits for the user to make the selection. ***The Select method displays nothing, if the Items collection is empty***. The Select property returns the identifier of the item being clicked. If no item has been clicked (or the user clicked outside of the context menu), the Select property returns 0. The Items property gives accesses to the Items collection of the control, so you can add, remove or update the items to be displayed. The Add method adds a new item to the Items collection. The ToString property loads or saves the control items from a string.

In case your context menu displays check-boxes, radio buttons, or items with Edit fields inside, you can use the GetChecked property gets a collection of checked items. The GetUnchecked property gets a collection of checked items. The GetRadio method gets a safe array with the radio-items being checked within a radio group. Also, you can use the Get method to retrieve a collection of Item objects based on your criteria. The control fires the SelectItem event when the user clicks an item.

The following samples show how to create the context menu, add a few items and call the Select method:

VB6, VBA (MS Access, Excell...), VB.NET for /COM

```
With Pivot1
   With CreateObject("Exontrol.ContextMenu")
      .Items.ToString = "Item A,Item B,Item C"
      Debug.Print( .Select() )
   End With
End With
```

VB.NET

```
With Expivot1
   ' Add 'exontrol.excontextmenu.dll' reference to your project.
   With New exontrol.EXCONTEXTMENULib.excontextmenu()
      .Items.ToString = "Item A,Item B,Item C"
      Debug.Print( .Select() )
   End With
End With
```

C++

```
   /*
      Includes the definition for CreateObject function like follows:
      #include <comdef.h>
      IUnknownPtr CreateObject( BSTR Object )
      {
         IUnknownPtr spResult;
         spResult.CreateInstance( Object );
         return spResult;
      };
   */
   /*
      Copy and paste the following directives to your header file as
      it defines the namespace 'EXCONTEXTMENULib' for the library: 'ExContextMenu
1.0 Type Library'
      #import <ExContextMenu.dll>
      using namespace EXCONTEXTMENULib;
   */
   EXCONTEXTMENULib::IExContextMenuPtr var_ExContextMenu =
::CreateObject(L"Exontrol.ContextMenu");
      var_ExContextMenu->GetItems()->PutToString(L"Item A,Item B,Item C");
      OutputDebugStringW( _bstr_t(var_ExContextMenu-
>Select(vtMissing,vtMissing,vtMissing)) );
```

C++ Builder

```
/*
   Select the Component\Import Component...\Import a Type Library,
   to import the following Type Library:
        ExContextMenu 1.0 Type Library
    TypeLib: e:\Exontrol\ExContextMenu\project\Site\ExContextMenu.dll
    to define the namespace: Excontextmenulib_tlb
*/
//#include "EXCONTEXTMENULIB_TLB.h"
Excontextmenulib_tlb::IExContextMenuPtr var_ExContextMenu =
Variant::CreateObject(L"Exontrol.ContextMenu");
    var_ExContextMenu->Items->ToString = L"Item A,Item B,Item C";
    OutputDebugString( PChar(var_ExContextMenu-
>Select(TNoParam(),TNoParam(),TNoParam())) );
```

C#

```
// Add 'exontrol.excontextmenu.dll' reference to your project.
exontrol.EXCONTEXTMENULib.excontextmenu var_ExContextMenu = new
exontrol.EXCONTEXTMENULib.excontextmenu();
    var_ExContextMenu.Items.ToString = "Item A,Item B,Item C";
    System.Diagnostics.Debug.Print( var_ExContextMenu.Select(null,null,null).ToString()
);
```

X++ (Dynamics Ax 2009)

```
    COM com_ExContextMenu,com_Items;
    anytype var_ExContextMenu,var_Items;
    ;
    // Add 'excontextmenu.dll' reference to your project.
    // Add 'ExContextMenu 1.0 Type Library' reference to your project.
    var_ExContextMenu = COM::createFromObject(new
EXCONTEXTMENULib.excontextmenu()); com_ExContextMenu = var_ExContextMenu;
        var_Items = COM::createFromObject(com_ExContextMenu.Items()); com_Items =
var_Items;
        com_Items.ToString("Item A,Item B,Item C");
        print( com_ExContextMenu.Select() )
```

Delphi 8 (.NET only)

```
with (ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMenu'))
as EXCONTEXTMENULib.ExContextMenu) do
```

```
begin
   Items.ToString := 'Item A,Item B,Item C';
   OutputDebugString( Select(Nil,Nil,Nil) );
end;
```

Delphi (standard)

```
with
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMen
 as EXCONTEXTMENULib_TLB.ExContextMenu) do
begin
   Items.ToString := 'Item A,Item B,Item C';
   OutputDebugString( Select(Null,Null,Null) );
end;
```

VFP

```
with CreateObject("Exontrol.ContextMenu")
   .Items.ToString = "Item A,Item B,Item C"
   DEBUGOUT( .Select() )
endwith
```

property ExContextMenu.SelForeColor as Color

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

Type	Description
Color	A Color expression that specifies the forground color to show the selected / highlighted item.

The SelForeColor property specifies the foreground color of the item being selected / highlighted. The SelBackColor property specifies the visual appearance of the item being selected / highlighted. The ForeColor property specifies the control's foreground color. The BackColor property specifies the control's background color. The Background property specifies the visual appearance for different parts of the control. The Appearance property specifies the menu's frame appearance.

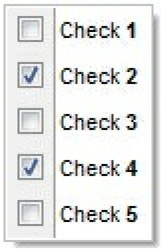
property ExContextMenu.ShowCheckedAsSelected as ShowCheckedAsSelectedEnum

Specifies whether the checked items shows as selected.

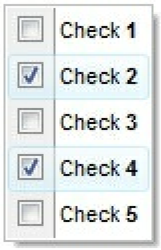
Type	Description
ShowCheckedAsSelectedEnum	A ShowCheckedAsSelectedEnum expression that specifies whether the checked items show as selected.

By default, the ShowCheckedAsSelected property is exDisplayItemCheckDefault. Use the ShowCheckedAsSelected property on non zero, to show the checked items as selected. A checked item is an item with the Check or Radio property set on True and the Checked property is True. The SelBackColor property indicates the color to show background of the selected / highlighted item. The AllowToggleRadio property on True, allows a radio button to set on zero (unchecked), if the user clicks twice the radio button. The ShowCheckedAsSelected property of the Item object specifies whether the individual checked item is shown as selected. The ShowCheckedAsSelectedTransparency property specifies the transparency (percent) to show the checked items when selected.

The following screen shot shows the control when the ShowCheckedAsSelected property is exDisplayItemCheckDefault(by default):



The following screen shot shows the control when the ShowCheckedAsSelected property is exDisplayItemCheckHighlight:



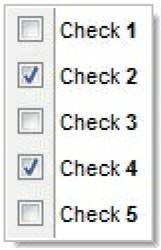
property ExContextMenu.ShowCheckedAsSelectedTransparency as Long

Specifies the transparency (percent) to show the checked items when selected.

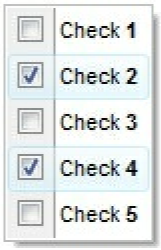
Type	Description
Long	A Long expression that specifies the transparency (percent) to show the checked items when selected. The valid values are from 0 (opaque), to 100 (fully transparent).

By default, the ShowCheckedAsSelectedTransparency property is 50 (semi-transparent) The ShowCheckedAsSelectedTransparency property specifies the transparency (percent) to show the checked items when selected. Use the ShowCheckedAsSelected property on non zero, to show the checked items as selected. A checked item is an item with the Check or Radio property set on True and the Checked property is True. The SelBackColor property indicates the color to show background of the selected / highlighted item. The AllowToggleRadio property on True, allows a radio button to set on zero (unchecked), if the user clicks twice the radio button. The ShowCheckedAsSelected property of the Item object specifies whether the individual checked item is shown as selected.

The following screen shot shows the control when the ShowCheckedAsSelected property is exDisplayItemCheckDefault(by default):



The following screen shot shows the control when the ShowCheckedAsSelected property is exDisplayItemCheckHighlight:

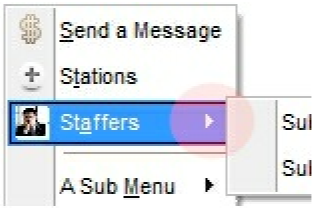


property ExContextMenu.ShowPopupArrow(ItemHighlited as Boolean) as ShowPopupArrowEnum

Indicates the type of the arrow to be shown when the item displays the sub-menu.

Type	Description
ItemHighlited as Boolean	A Boolean expression that specifies whether the arrow is shown to the selected/highlighted item. *In VFP, you should use 0 or 1, instead .F. or .T.*
ShowPopupArrowEnum	A ShowPopupArrowEnum expression that specifies the arrow to be shown on an item that displays a submenu.

By default, the ShowPopupArrow(True) property is exShowPopupArrowLight, and the ShowPopupArrow(False) property is exShowPopupArrowDark. In other words, when the item is selected/highlighted a light arrow is displayed, while when the item it is not selected/highlighted a dark arrow is displayed. Use the ShowPopupArrow property to specify how the item with sub-menu displays the popup arrow.



property ExContextMenu.ShowPopupEffect as ShowPopupEffectEnum

Specifies the effect to show the popup menu when clicking an item, such as scrolling, lighting up, and so on.

Type	Description
ShowPopupEffectEnum	A ShowPopupEffectEnum expression that specifies effect to be applied when the user opens the menu or a sub-menu.

By default, ShowPopupEffect property is exShowPopupLightUp. Use the ShowPopupEffect property to disable the effect to be applied when the user opens the menu or any sub-menu.

property ExContextMenu.Template as String

Specifies the control's template.

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

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

For instance, the following sample adds a few items and displays the context menu:

```
Set n = New EXCONTEXTMENULib.ExContextMenu
n.Template = "Items.ToString = `Item A[id=1001],Item B,Item C,Item D`;Select()"
```

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.

The Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as a progID
<text> any string of characters

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 ExContextMenu.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.

The Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as a progID
<text> any string of characters

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 ExContextMenu.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 ExContextMenu.ToolTipDelay as Long

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

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

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. 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 ToolTip property to assign a tooltip to an item. Use the ToolTipFont property or <font> HTML element to assign a new font for tooltips.

property ExContextMenu.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

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

property ExContextMenu.ToolTipPopDelay as Long

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

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

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

property ExContextMenu.ToolTipWidth as Long

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

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

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

property ExContextMenu.ToString as String

Loads or saves the Items collection using string representation (shortcut of Items.ToString property).

Type	Description
String	A String expression that specifies the items to be added. The list of items is separated by , (comma) character, while sub-menus are include between () parenthesis. The [] brackets indicates the options to be applied on the item

The ToString property of the control is equivalent with the ToString property of the Items object.

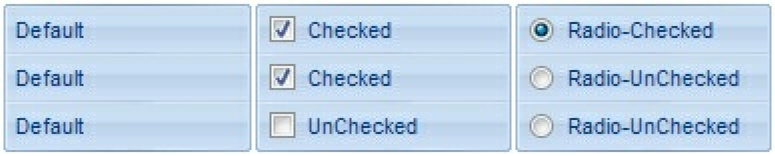
The ToString syntax in BNF notation:

<ToString> ::= <ITEMS>
<ITEMS> ::= <ITEM>["("<ITEMS>")"][","<ITEMS>]
<ITEM> ::= <CAPTION>[<OPTIONS>]
<OPTIONS> ::= "["<OPTION>"]"["["<OPTIONS>"]"]
<OPTION> ::= <PROPERTY>["="<VALUE>]
<PROPERTY> ::= "img" | "himg" | "sep" | "id" | "typ" | "group" | "chk" | "button" | "align" | "spchk" | "show" | "rad" | "dis" | "showdis" | "bld" | "itl" | "stk" | "und" | "bg" | "fg" | "edittype" | "edit" | "mask" | "border" | "editwidth" | "captionwidth" | "height" | "grp" | "tfi" | "ttp" | "min" | "max" | "tick" | "freq" | "ticklabel" | "small" | "large" | "spin" | "ettp" | "float" | "close" | "local" | "popupapp" | "itemspad" | "itemsbg" | "itemsbghot" | "itemsbgext" | "visible" | "tab" | "pad" | "bghot" | "bgsel" | "bgselhot" | "arrow" | "popupalign" | "popupoffset" | "popupat"

where the <CAPTION> is the HTML caption to be shown on the context menu item. The <VALUE> indicates the value of giving property.

- id=<VALUE>, where <VALUE> is an integer expression, that indicates the identifier of the item.
- bg=<VALUE>, specifies the item's background color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- bghot=<VALUE>, specifies the item's background color, while the cursor hovers the item, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- bgsel=<VALUE>, specifies the item's background color, while the item is checked/selected, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.

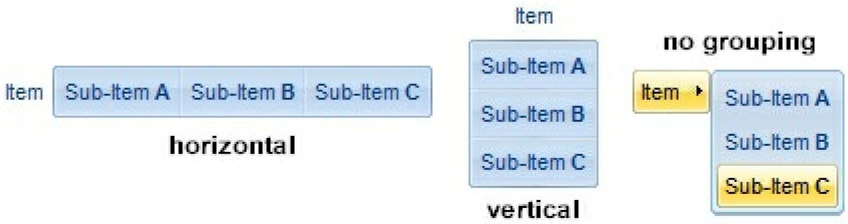
- bgselhot=<VALUE>, specifies the item's background color, while the item is checked/selected and the cursor hovers it, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- fg=<VALUE>, specifies the item's foreground color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or a integer expression.
- sep, specifies an separator item
- dis, specifies a disabled item
- showdis=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the item shows as disabled, but it is still enabled
- bld, specifies that the item appears in bold
- itl, specifies that the item appears in italics
- stk, specifies that the item appears as strikeout
- und, specifies that the item is underlined
- align=<VALUE>, where <VALUE> could be one of the following:
  - **0** (left), to align the item's caption to the left
  - **1** (center), to center the item's caption
  - **2** (right), to align the item's caption to the right
- captionwidth=<VALUE>, specifies the width to show the HTML caption of the item. where <VALUE> could be a integer expression. A negative value indicates that no limitation is applied to the item's caption, so no truncate caption is shown
- height=<VALUE>, specifies the height to show the item, where <VALUE> could be a positive integer expression
- pad=<VALUE>, specifies the padding (space between the menu border and the item content) to display the item. The <VALUE> is a list of coordinates such as left,top,right,bottom
- img=<VALUE>, where <VALUE> is an integer expression, that indicates the index of the icon being displayed for the item.
- himg=<VALUE>, where <VALUE> indicates the key of the picture to be displayed for the item.



- typ=<VALUE>, where <VALUE> could be one of the following:
  - **0** for default/regular items (no check/radio button is associated with the item),
  - **1** for items that display a check/box (chk),
  - **2** to display radio buttons (rad)
- chk[=<VALUE>], where <VALUE> could be **0** for unchecked, or **not zero** for checked. The chk option makes the item to display a check box. If the <VALUE> is missing the item still displays an un-checked check box.

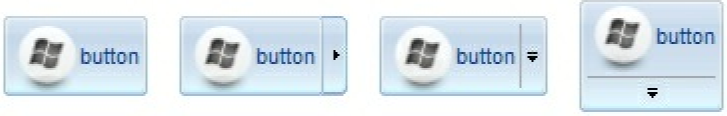
- rad=<VALUE>, where <VALUE> could be **0** for unchecked radio button or **not zero** to for checked radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored.
- grp=<VALUE>, defines the radio group. It should be used when you define more groups of radio buttons. A group of radio buttons means that only one item could be checked at one time. The rad option specifies that the item displays a radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored. The <VALUE> could be any integer expression.

- show=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the checked item shows as selected
- spchk=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the item's sub menu is shown only if the item is checked.

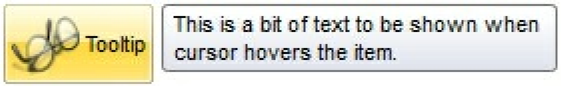


- group=<VALUE>, where <VALUE> could be a bit-or combination (+) of the following values:
  - **0** (exNoGroupPopup), No grouping is performed on the sub-menu, so the sub-items are shown to a float popup,
  - **1** (exGroupPopup), Groups and displays the sub-menu items on the current item, arranged from left to right/horizontally
  - **2** (exNoGroupPopupFrame), Prevents showing the frame around each grouping item.
  - **4** (exGroupPopupCenter), Shows the grouping popup aligned to the center of the current item.
  - **8** (exGroupPopupRight), Shows the grouping popup aligned to the right of the current item.
  - **16** (exGroupPopupEqualWidth), Shows the items that make the group of the same width
  - **32** (exGroupPopupEqualHeight), Shows the items that make the group of the same height
  - **64** (exGroupPopupFrameSolidBox), Shows a solid frame around the grouped

    items
    - **128** (exGroupPopupFrameThickBox), Shows a solid thick-frame around the grouped items
    - **256** (exGroupPopupVertical), Groups and displays the sub-menu items on the current item, arranged from top to bottom/vertically

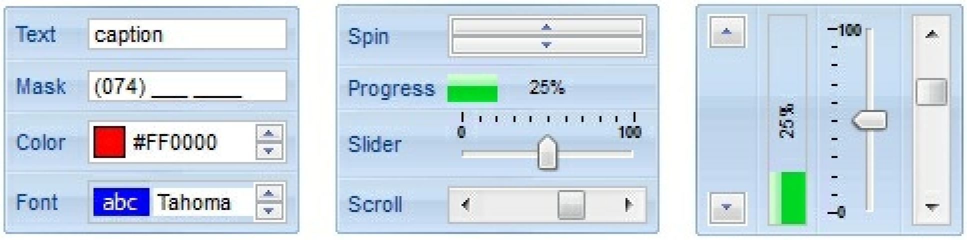


- button=<VALUE>, where <VALUE> could be a bit-or combination (+) of the following values.
    - **0** (exShowAsButtonNone), No button is shown,
    - **1** (exShowAsButton), Shows the item as a button
    - **2** (exShowAsButtonAutoSize), Fits the button to cover the item's caption instead showing on the entire item
    - **17** (exShowAsSelectButton), Shows the item as a select button, which is composed by two-fields, one indicates the default button, while the second field specifies the drop down button that displays the items in the current's sub-menu collection. The drop down button is shown to the right-side of the default button. The item must have a submenu, else no drop down is displayed.
    - **273** (exShowAsSelectButtonBottom), Shows the item as a select button, which is composed by two-fields, one indicates the default button, while the second field specifies the drop down button that displays the items in the current's sub-menu collection. The drop down button is shown to the bottom-side of the default button. The item must have a submenu, else no drop down is displayed.



This is a bit of text to be shown when cursor hovers the item.

- ttp=<VALUE>, defines the item's tooltip. The <VALUE> could be any HTML string expression. The item's tooltip is shown when the user hovers the item.



Text caption
Mask (074) ___ ____
Color #FF0000
Font abc Tahoma
Spin
Progress 25%
Slider 0 100
Scroll

- edittype=<VALUE>, associates an edit field to the item, where <VALUE> could be a combination of one or more of the following values:
    - **0** (exItemDisableEdit), No editor is assigned to the current item.
    - **1** (exItemEditText), A text-box editor is assigned to the current item.
    - **2** (exItemEditMask), A masked text-box editor is assigned to the current item.

  - **3** (exItemEditSlider), A slider editor is assigned to the current item. This can be combined with 1024.
  - **4** (exItemEditProgress), A progress editor is assigned to the current item. This can be combined with 1024.
  - 5 (exItemEditScrollBar), A scrollbar editor is assigned to the current item. This can be combined with 1024.
  - **6** (exItemEditColor), A color editor is assigned to the current item.
  - 7 (exItemEditFont), A font editor is assigned to the current item.
  - 256 (exItemEditReadOnly), specifies that the item's editor is shown as disabled. This value could be combined with one of the values from 0 to 7 or 512
  - 512 (exItemEditSpin), A spin editor is assigned to the current item. This value could be combined with one of the values from 0 to 7 or 256
  - 1024 (exItemEditVertical), The editor is shown vertically rather than horizontally. This value has effect for exItemEditSlider, exItemEditProgress or exItemEditScrollBar
- edit=<VALUE>, specifies the caption to be shown in the item's edit field, where <VALUE> could be any string
- mask=<VALUE>, specifies the mask to be applied on a masked editor. This option is valid for exItemEditMask edit. Use the float option to allow masking floating point numbers. See Masking for more information about <VALUE> of the mask option. See Masking Float for more information about <VALUE> if the float option is used.
- float=<VALUE>, Specifies whether the mask field masks a floating point number. This option is valid for exItemEditMask edit. See Masking Float for more information about <VALUE> of mask option, if the float option is used. The <VALUE> could be **0** for standard masking field or **not zero** to specify that the field is masking a floating point.
- border=<VALUE>, specifies the border to be shown on the item's edit field, where <VALUE> could be one of the following:
  - **0** (exEditBorderNone), No border is shown.
  - **-1** (exEditBorderInset), shows an inset border
  - **1** (exEditBorderSingle), shows a frame border
- editwidth=<VALUE>, specifies the width to show the edit field inside the item, where <VALUE> could be a integer expression. A negative value indicates that the field goes to the end of the item
- min=<VALUE>, defines the minimum value of the edit field. The <VALUE> could be any integer expression, and specifies the minimum value for any slider, progress, scroll, spin, or range editor.
- max=<VALUE>, defines the maximum value of the edit field. The <VALUE> could be any integer expression, and specifies the maximum value for any slider, progress, scroll, spin, or range editor.
- tick=<VALUE>, defines where the ticks of the slider edit appear. This option is valid for exItemEditSlider edit. The <VALUE> could be one of the following values:
  - **0** (exBottomRight), The ticks are displayed on the bottom/right side.
  - **1** (exTopLeft), The ticks are displayed on the top/left side.

    - **2** (exBoth), The ticks are displayed on the both side.
    - **3** (exNoTicks), No ticks are displayed.
- freq=<VALUE>, indicates the ratio of ticks on the slider edit. This option is valid for exItemEditSlider edit. The <VALUE> could be a positive integer expression.
- ticklabel=<VALUE>, indicates the HTML label to be displayed on slider's ticks. This option is valid for exItemEditSlider edit. See Tick Label Expression for more information about <VALUE> of the ticklabel option.
- small=<VALUE>, indicates the amount by which the edit's position changes when the user presses the arrow key (left, right, or button). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive integer expression.
- large=<VALUE>, indicates the amount by which the edit's position changes when the user presses the CTRL + arrow key (CTRL + left, CTRL + right). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive integer expression.
- spin=<VALUE>, specifies the step to advance when user clicks the editor's spin.. This option is valid for exItemEditSpin edit. The <VALUE> could be a positive integer expression.
- ettp=<VALUE>, specifies the HTML tooltip to be shown when the item's value is changed. This option is valid for exItemEditSlider/exItemEditScrollBar edit. The <VALUE> could be any string expression, including built-in HTML tags

- arrow=<VALUE>. The <VALUE> could be **0** for hiding the arrow or **not zero** to show the arrow. Indicates whether an item that has a sub-menu shows or hides its popup arrow. If the <VALUE> is missing, the item shows no arrow.
- local=<VALUE>. The <VALUE> could be **0** for standard popup or **not zero** to specify that the field is a local popup. Specifies whether the item's popup is shown as local. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus.
- close=<VALUE>, Specifies the way the hosting menu is closed when the user clicks the item. If the close flag is missing, the <VALUE> is 3 (exCloseOnNonClickable), by default. The <VALUE> could be one of the following values:
    - **0** (exCloseOnClick), The popup menu is closing when the user clicks the item.
    - **1** (exCloseOnDblClick), The popup menu is closing when the user double clicks the item.
    - **2** (exCloseOnClickOutside), The popup menu is closing when the user clicks outside of the menu.
    - **3** (exCloseOnNonClickable), The popup menu is closing when the user clicks a non-clickable item (regular items). The non-clickable items is any item that's not a separator, popup, disabled or check or radio items, clicking a check-box item will makes the check box to change its state instead closing the context menu.
- popupapp=<VALUE> indicates the visual appearance of the item's submenu when the popup is shown. The <VALUE> could be a predefine value like shown bellow, or an

integer expression that refers an EBN object.

  - **0** (NoBorder)
  - **1** (FlatBorder)
  - **2** (SunkenBorder)
  - **3** (RaisedBorder)
  - **4** (EtchedBorder)
  - **5** (BumpBorder)
  - **6** (ShadowBorder)
  - **7** (InsetBorder)
  - **8** (SingleBorder)
- itemsbg=<VALUE>, specifies the items background color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- itemsbghot=<VALUE>, specifies the items background color, while the cursor hovers the items, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- popupalign=<VALUE>, Indicates how the item's sub-menu is aligned relative to the parent item. The popupalign has no effect for an item that displays a select- button. The <VALUE> could be a combination of one or more of the following values:
  - **0** (exShowPopupAlignNone), The popup menu is shown on top of the item, aligned to the left (no down and right, so up and left)
  - **1** (exShowPopupAlignDown), The popup menu is shown down. If missing, the popup menu is shown up.
  - **2** (exShowPopupAlignRight), The popup menu is shown aligned to the right, else if missing, the popup menu is shown aligned to the left.
- popupat=<VALUE>, specifies the identifier of the item where the current item's submenu/popup is displayed. The <VALUE> could be any integer expression. If there is no identifier with giving value, the option has no effect.
- popupoffset=<VALUE>, specifies the offset (horizontal,vertical) to display the item's submenu/popup relative to its default position.
- itemspad=<VALUE>, specifies the padding (space between the menu border and the item content) to display the items. The <VALUE> is a list of coordinates such as left,top,right,bottom
- visible=<VALUE>, specifies the maximum number of visible items at one time, where the <VALUE> could be any integer expression.
- tab=<VALUE>, specifies the identifier of the item/tab where the current group-popup is shown instead. The <VALUE> could be any integer expression. If there is no identifier with giving value, the option has no effect.
- itemsbgext=<VALUE>, indicates additional colors, text, images that can be displayed on the items background using the EBN String Format. The <VALUE> should be in EBN

String Format. For instance, *[itemsbgext=bottom[2],bottom[16,text=`</fgcolor><fgcolor 6D6AAA>Views</fgcolor><fgcolor A0A0A0>`,align=0x21]]*, shows the Views aligned to the bottom, with a different foreground color.

Masking, (mask option)

For instance, the following input-mask (ext-phone)

!(999) 000 0000;1;;select=1,empty,overtype,warning=invalid character,invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field."

indicates the following:

- The pattern should contain 3 optional digits *999*, and 7 required digits *000 0000*, aligned to the right, *!*.
- The second part of the input mask indicates *1*, which means that all literals are included when the user leaves the field.
- The entire field is selected when it receives the focus, *select=1*
- The field supports *empty* value, so the user can leave the field with no content
- The field enters in *overtype* mode, and insert-type mode is not allowed when user pressed the Insert key
- If the user enters any invalid character, a *warning* tooltip with the message "*invalid character*" is displayed.
- If the user tries to leave the field, while the field is not validated (all 7 required digits completed), the *invalid* tooltip is shown with the message "*The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field.*" The *<%mask%>* is replaced with the first part of the input mask *!(999) 000 0000*

The four parts of an input mask, or the Mask property supports up to four parts, separated by a semicolon (;). For instance, "`Time: `00:00:00;;0;overtype,warning=<fgcolor FF0000>invalid character,beep", indicates the pattern "00:00" with the prefix Time:, the masking character being the 0, instead _, the field enters in over-type mode, insert-type mode is not allowed, and the field beeps and displays a tooltip in red with the message invalid character when the user enters an invalid character.

Input masks are made up one mandatory part and three optional parts, and each part is separated by a semicolon (;). If a part should use the semicolon (;) it must uses the \; instead

The purpose of each part is as follows:

1. The first part (pattern) is mandatory. It includes the mask characters or string (series

of characters) along with placeholders and literal data such as, parentheses, periods, and hyphens.

The following table lists the placeholder and literal characters for an input mask and explains how it controls data entry:

- ***#**, a digit, +, - or space (entry not required).*
- ***0**, a digit (0 through 9, entry required; plus [+] and minus [-] signs not allowed).*
- ***9**, a digit or space (entry not required; plus and minus signs not allowed).*
- ***x**, a lower case hexa character, [0-9],[a-f] (entry required)*
- ***X**, an upper case hexa character, [0-9],[A-F] (entry required)*
- ***A**, any letter, digit (entry required).*
- ***a**, any letter, digit or space (entry optional).*
- ***L**, any letter (entry require).*
- ***?**, any letter or space (entry optional).*
- ***&**, any character or a space (entry required).*
- ***C**, any character or a space (entry optional).*
- ***>**, any letter, converted to uppercase (entry required).*
- ***<**, any letter, converted to lowercase (entry required).*
- *, any characters combinations
- ***{ min,max }** (Range), indicates a number range. The syntax {min,max} (Range), masks a number in the giving range. The min and max values should be positive integers. For instance the mask {0,255} masks any number between 0 and 255.*
- ***[...]** (Alternative), masks any characters that are contained in the [] brackets. For instance, the [abcdA-D] mask any character: a,b,c,d,A,B,C,D*
- \, indicates the escape character
- ***ť**, (ALT + 175) causes the characters that follow to be converted to uppercase, until Ť(ALT + 174) is found.*
- ***Ť**, (ALT + 174) causes the characters that follow to be converted to lowercase, until ť(ALT + 175) is found.*
- ***!**, causes the input mask to fill from right to left instead of from left to right.*

Characters enclosed in double quotation ("" or ``) marks will be displayed literally. If this part should display/use the semicolon (;) character is should be included between double quotation ("" or ``) characters or as \; (escape).

2. The second part is optional and refers to the embedded mask characters and how they are stored within the field. If the second part is set to 0 (default, exClipModeLiteralsNone), all characters are stored with the data, and if it is set to 1

(exClipModeLiteralsInclude), the literals are stored, not including the masking/placeholder characters, if 2 (exClipModeLiteralsExclude), just typed characters are stored, if 3(exClipModeLiteralsEscape), optional, required, editable and escaped entities are included. No double quoted text is included.

3. The third part of the input mask is also optional and indicates a single character or space that is used as a placeholder. By default, the field uses the underscore (_). If you want to use another character, enter it in the third part of your mask. Only the first character is considered. If this part should display/use the semicolon (;) character is should be \; (escape)
4. The forth part of the input, indicates a list of options that can be applied to input mask, separated by comma(,) character.

   The known options for the forth part are:

   - ***float****, indicates that the field is edited as a decimal number, integer. The first part of the input mask specifies the pattern to be used for grouping and decimal separators, and - if negative numbers are supported. If the first part is empty, the float is formatted as indicated by current regional settings. For instance, "##;;;float" specifies a 2 digit number in float format. The grouping, decimal, negative and digits options are valid if the float option is present.*

   - ***grouping****=value, Character used to separate groups of digits to the left of the decimal. Valid only if float is present. For instance ";;;float,grouping=" indicates that no grouping is applied to the decimal number (LOCALE_STHOUSAND)*
   - ***decimal****=value, Character used for the decimal separator. Valid only if float is present. For instance ";;;float,grouping= ,decimal=\," indicates that the decimal number uses the space for grouping digits to the left, while for decimal separator the comma character is used (LOCALE_SDECIMAL)*
   - ***negative****=value, indicates whether the decimal number supports negative numbers. The value should be 0 or 1. 1 means negative numbers are allowed. Else 0 or missing, the negative numbers are not accepted. Valid only if float is present.*
   - ***digits****=value, indicates the max number of fractional digits placed after the decimal separator. Valid only if float is present. For instance, ";;;float,digits=4" indicates a max 4 digits after decimal separator (LOCALE_IDIGITS)*
   - ***password****[=value], displays a black circle for any shown character. For instance, ";;;password", specifies that the field to be displayed as a password. If the value parameter is present, the first character in the value indicates the password character to be used. By default, the * password character is used for non-TrueType fonts, else the black circle character is used. For instance, ";;;password=*", specifies that the field to be displayed as a password, and use*

*the * for password character. If the value parameter is missing, the default password character is used.*

- ***right**, aligns the characters to the right. For instance, "(999) 999-9999;;;right" displays and masks a telephone number aligned to the right. **readonly**, the editor is locked, user can not update the content, the caret is available, so user can copy the text, excepts the password fields.*
- ***inserttype**, indicates that the field enters in insert-type mode, if this is the first option found. If the forth part includes also the overtype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;inserttype,overtype", indicates that the field enter in insert-type mode, and over-type mode is allowed. The "##:##;;0;inserttype", indicates that the field enter in insert-type mode, and over-type mode is not allowed.*
- ***overtype**, indicates that the field enters in over-type mode, if this is the first option found. If the forth part includes also the inserttype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;overtype,inserttype", indicates that the field enter in over-type mode, and insert-type mode is allowed. The "##:##;;0;overtype", indicates that the field enter in over-type mode, and insert-type mode is not allowed.*
- ***nocontext**, indicates that the field provides no context menu when user right clicks the field. For instance, ";;;password,nocontext" displays a password field, where the user can not invoke the default context menu, usually when a right click occurs.*
- ***beep**, indicates whether a beep is played once the user enters an invalid character. For instance, "00:00;;;beep" plays a beep once the user types in invalid character, in this case any character that's not a digit.*
- ***warning**=value, indicates the html message to be shown when the user enters an invalid character. For instance, "00:00:00;;;warning=invalid character" displays a "invalid character" tooltip once the user types in invalid character, in this case any character that's not a digit. The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape)*
- ***invalid**=value, indicates the html message to be displayed when the user enters an inappropriate value for the field. If the value is missing or empty, the option has no effect, so no validation is performed. If the value is a not-empty value, the validation is performed. If the value is single space, no message is displayed and the field is keep opened while the value is inappropriate. For instance, "!(999) 000 0000;;;invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field." displays the "The value you entered isn't appropriate for the input mask '...' specified for this field." tooltip once the user leaves the field and it is not-valid (for instance, the field includes*

entities required and uncompleted). The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape). This option can be combined with empty, validateas.

- ***validateas**=value, specifies the additional validation is done for the current field. If value is missing or 0 (exValidateAsNone), the option has no effect. The validateas option has effect only if the invalid option specifies a not-empty value. Currently, the value can be 1 (exValidateAsDate), which indicates that the field is validated as a date. For instance, having the mask "!00/00/0000;;0;empty,validateas=1,invalid=Invalid date!,warning=Invalid character!,select=4,overtype", indicates that the field is validate as date (validateas=1).*
- ***empty**, indicates whether the field supports empty values. This option can be used with invalid flag, which indicates that the user can leave the field if it is empty. If empty flag is present, the field displays nothing if no entity is completed (empty). Once the user starts typing characters the current mask is displayed. For instance, having the mask "!(999) 000 0000;;;empty,select=4,overtype,invalid=invalid phone number,beep", it specifies an empty or valid phone to be entered.*
- ***select**=value, indicates what to select from the field when it got the focus. The value could be 0 (nothing, exSelectNoGotFocus), 1 (select all, exSelectAllGotFocus), 2 (select the first empty and editable entity of the field, exSelectEditableGotFocus), 3 (moves the cursor to the beginning of the first empty and editable entity of the field, exMoveEditableGotFocus), 4 (select the first empty, required and editable entity of the field, exSelectRequiredEditableGotFocus), 5 (moves the cursor to the beginning of the first empty, required and editable entity of the field, exMoveRequiredEditableGotFocus). For modes 2 and 4 the entire field is selected if no matching entity is found. For instance, "`Time:`XX:XX;;;select=1" indicates that the entire field (including the Time: prefix) is selected once it get the focus. The "`Time:`XX:XX;;;select=3", moves the cursor to first X, if empty, the second if empty, and so on*

Experimental:
***multiline**, specifies that the field supports multiple lines.*
***rich**, specifies that the field displays a rich type editor. By default, the standard edit field is shown*
***disabled**, shows as disabled the field.*

Masking-Float, (mask, float option)

The [mask=<VALUE>] property may indicate the followings, if the [float=-1] is present

- **negative number**: if the first character in the mask is - (minus) the control supports negative numbers. Pressing the - key will toggle the sign of the number. The + sign is never displayed.
- **decimal symbol**: the last character that's different than # (digit), or 0 (zero) indicates the decimal symbol. If it is not present the control mask a floating point number without decimals.
- **thousand symbol**: the thousand symbol is the last character that's not a # (digit), 0 (zero) or it is not the decimal symbol as explained earlier, if present.
- the maximum **number of decimals** in the number (the # or 0 character after the decimal symbol)
- the maximum number of digits in the integer part (the number of # or 0 character before decimal symbol)
- the **0** character indicates **a leading-zero**. The count of 0 (zero) characters before decimal character indicates the leading-zero for integer part of the control, while the count of 0 (zero) characters after the decimal separator indicates the leading-zero for decimal part of the control. For instance, the Mask on "-###,###,##0.00", while the control's Text property is 1, the control displays 1.00, if 1.1 if displays 1.10, and if empty, the 0.00 is displayed.

If the <VALUE> property is empty, the control takes the settings for the regional options like: Decimal Symbol , No. of digits after decimal, Digit grouping symbol.

Here are few samples:

The <VALUE>"**-###.###.##0,00**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator). This format displays leading-zeros.

The <VALUE>"**-###.###.###,##**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator)

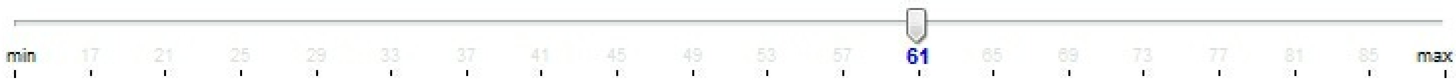
The <VALUE>"**-###,###,###.##**" filter floating point numbers a number for English settings ("." is the decimal sign, "," is the thousands separator)

The <VALUE>"**####**" indicates a max-4 digit number (positive) without a decimal symbol and without digit grouping

The <VALUE>"**-##.#**" filters a floating point number from the -99.9 to 99.9 ("." is the decimal sign, no thousands separator)

The <VALUE>"**#,###.##**" filters a floating point number from the 0 to 9,999.99 with digit grouping ("." is the decimal sign, "," is the thousands separator).

Tick Label Expression, (ticklabel option)



For instance:

- "value", shows the values for each tick.
- " (value=current ? '<font ;12><fgcolor=FF0000>' : '') + value", shows the current slider's position with a different color and font.
- "value = current ? value : ''", shows the value for the current tick only.
- "(value = current ? '<b><font ;10>' : '') + (value array 'ab bc cd de ef fg gh hi ij jk kl' split ' ')" displays different captions for slider's values.

The The <VALUE> of [ticklabel] option is a formatted expression which result may include the HTML tags.

The The <VALUE> of [ticklabel] option indicates a formatting expression that may use the following predefined keywords:

- **value** gets the slider's position to be displayed
- **current** gets the current slider's value.
- **vmin** gets the slider's minimum value.
- **vmax** gets the slider's maximum value.
- **smin** gets the slider's selection minimum value.
- **smax** gets the slider's selection maximum value.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- **/** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array** (at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

  "expression array (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')"* is equivalent with *"month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'O';10:'N';11:'D')"*.

- ***in** (include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression =*

44) or (expression = 13)". The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch** (switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
    - #4/1/2009#, from hours 06:00 AM to 12:00 PM
    - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
    - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

    - 0 - empty (not initialized)
    - 1 - null
    - 2 - short
    - 3 - long
    - 4 - float
    - 5 - double
    - 6 - currency
    - 7 - date
    - 8 - string
    - 9 - object
    - 10 - error
    - 11 - boolean
    - 12 - variant
    - 13 - any
    - 14 - decimal
    - 16 - char
    - 17 - byte
    - 18 - unsigned short
    - 19 - unsigned long
    - 20 - long on 64 bits
    - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format

MM/DD/YYYY HH:MM:SS.

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
  - *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
  - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
    - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
    - 1 - Negative sign, number; for example, -1.1
    - 2 - Negative sign, space, number; for example, - 1.1

        - 3 - Number, negative sign; for example, 1.1-
        - 4 - Number, space, negative sign; for example, 1.1 -
    - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.
- a **split** b, splits the a using the separator b, and returns an array. For instance, the "weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' **split** ' '" gets the weekday as string. This operator can be used with the array

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.

- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

The The <VALUE> of [ticklabel] option can display labels using the following built-in HTML tags:

- **<b></b>** displays the text in **bold**.
- **<i></i>** displays the text in *italics*.
- **<u></u>** underlines the text.
- **<s></s>** ~~Strike~~-through text
- **<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>** displays text with a specified foreground color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<bgcolor=RRGGBB></bgcolor>** displays text with a specified background color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<br>** a forced line-break
- **<solidline>** The next line shows a solid-line on top/bottom side. If has no effect for a single line caption.
- **<dotline>** The next line shows a dot-line on top/bottom side. If has no effect for a single line caption.
- **<upline>** The next line shows a solid/dot-line on top side. If has no effect for a single line caption.
- **<r>** Right aligns the text
- **<c>** Centers the text
- **<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, in UNICODE configuration. 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;

EBN String Format, (itemsbgext option)

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

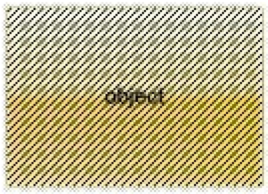
```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<client> ","] <attribute> [ "," <attributes> ] "]"
<client> ::= <expression> | <expression> "," <expression> "," <expression> ","
<expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> | <wordwrap> | <align> | <pattern> |
<patterncolor> | <frame> | <framethick> | <data> | <others>
<equal> ::= "="
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"  "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number> ")"
<string> ::= "`" <characters> "`" | "'" <characters> "'" | " <characters> "
```

```
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
<patterncolor> ::= "patterncolor" <equal> <color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

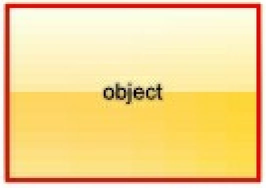
Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

Here's a few easy samples:

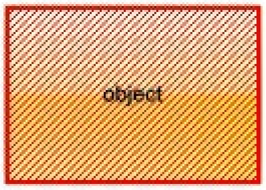
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



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

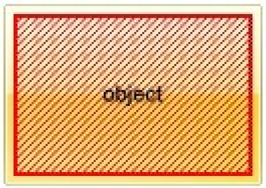


- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.

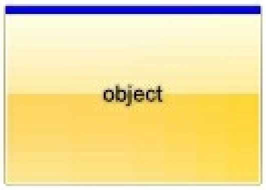


- "[[patterncolor=RGB(255,0,0)]
(none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25
draws a red thick-border around the object, with a patter inside, with a 4-pixels wide

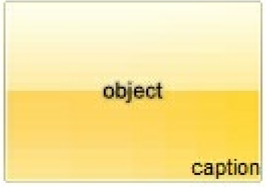
padding:



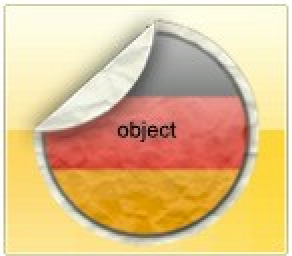
- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.



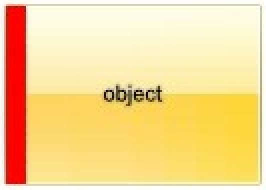
- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.



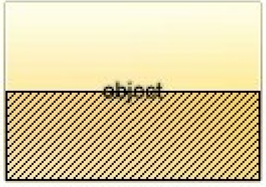
- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.



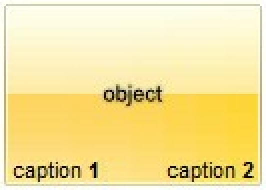
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



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



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



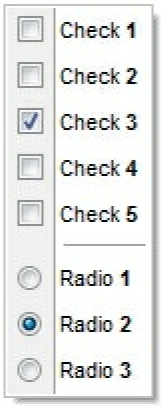
property ExContextMenu.UseVisualTheme as UIVisualThemeEnum

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

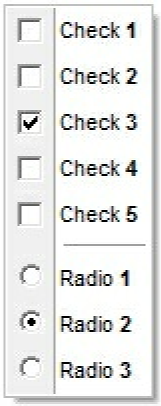
Type	Description
UIVisualThemeEnum	An UIVisualThemeEnum expression that specifies which UI parts of the control are shown using the current visual theme.

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

The following screen shot shows the control while the UseVisualTheme property is exDefaultVisualTheme:



since the second screen shot shows the same data as the UseVisualTheme property is exNoVisualTheme:



property ExContextMenu.Version as String

Retrieves the control's version.

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

The version property specifies the control's version.

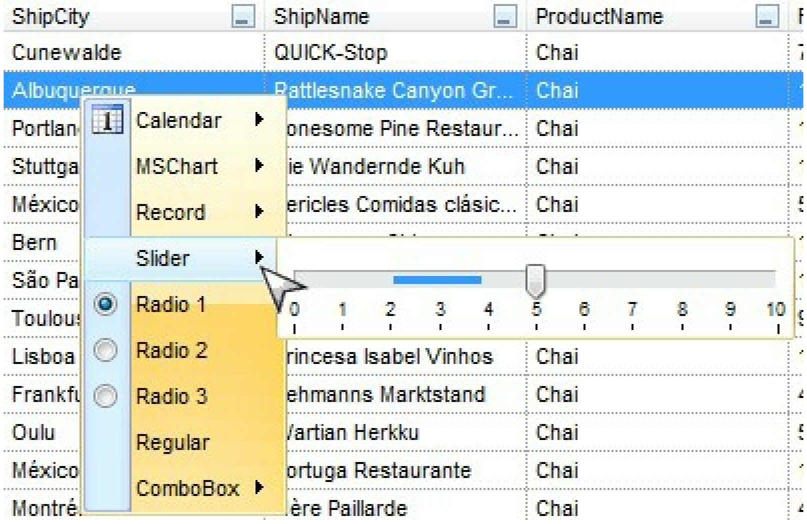
property ExContextMenu.Visibility as Long

Specify the popup's visibility in percents: 90% is barely visible, and 10% is nearly opaque.

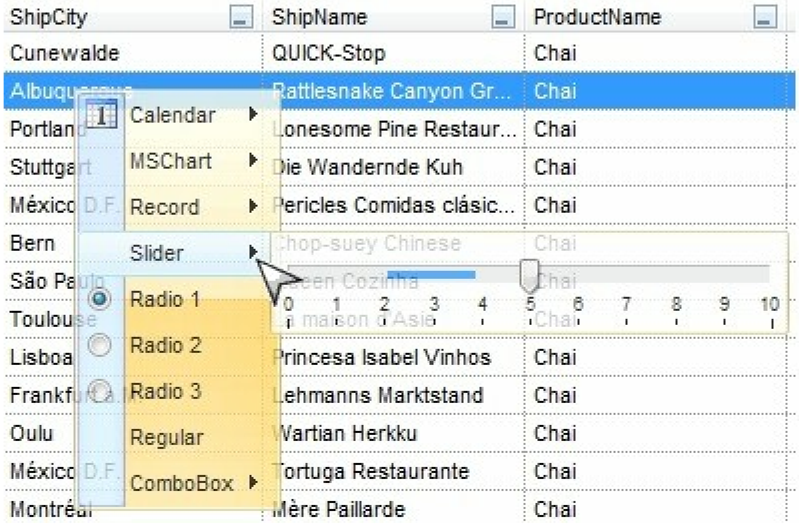
Type	Description
Long	A long expression that indicates the visibility of the popup menus.

By default, the Visibility is 100. Use the Visibility property to change the menu's visibility.

The following screen shot shows the menu when the Visibility is 100 (opaque, by default):



The following screen shot shows the menu when the Visibility is 80 (semi-transparent):



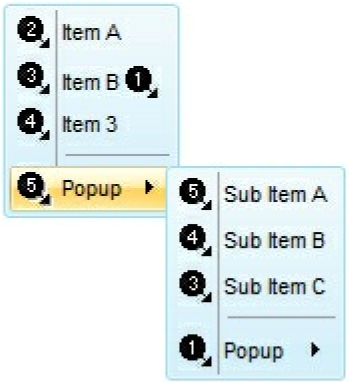
property ExContextMenu.VisualAppearance as Appearance

Retrieves the control's appearance.

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

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part. Use the Appearance property to change the menu's frame using an EBN skin object. The SelBackColor property specifies the visual appearance of the item being selected / highlighted. The Background property specifies the visual appearance for different parts of the control, including the radio-buttons, check-boxes or separator items.

The following screen shot shows the control's frame using a different EBN file:



Item object

The Item object holds information about an item in the context menu. The Item property searches recursively the item with giving identifier/caption. The Item object supports the following properties and methods:

Name	Description
Alignment	Retrieves or sets the item's caption alignment.
AllowEdit	Retrieves or sets a value indicating whether the item contains an edit control.
BackColor	Specifies the background color of the item.
Bold	Specifies whether the item's caption should appear in bold.
Caption	Retrieves or sets a value that indicates the item's caption.
CaptionWidth	Specifies the fixed width to display the item's caption.
Check	Retrieves or sets a value that indicates whether the item is of check type.
Checked	Retrieves or sets a value that indicates the item's state.
CloseOnClick	Specifies the way the owner menu is closed once the user clicks the item.
Cursor	Specifies the shape of the cursor when mouse hovers the item.
EditBorder	Specifies the border for the inside edit control.
EditCaption	Specifies the edit's caption when the item contains an edit control.
EditMask	Specifies the edit's mask when the item contains an masked edit control.
EditOption	Specifies different options for item's edit control.
EditValue	Specifies the edit's value when the item contains an edit control.
EditWidth	Specifies the width for the inside edit control.
Enabled	Retrieves or sets a value that indicates whether the item is enabled or disabled.
ForeColor	Specifies the foreground color of the item.
GroupPopup	Specifies whether the items of the sub-menu are grouped and displayed on the current item.

HotBackColor	Specifies the hot background color of the item (when the cursor hovers the item).
HTMLImage	Retrieves or sets a value that indicates the key of the image (HTMLPicture method) to be displayed on the item (left side).
ID	Retrieves or sets a value that specifies the item's identifier.
Image	Retrieves or sets a value that indicates the item's index image.
Italic	Specifies whether the item's caption should appear in italic.
ItemHeight	Specifies the fixed height to display the item.
Items	Retrieves an Items collection that indicates the item's sub menu. Retrieves Nothing, if the item contains no sub menu.
ItemType	Returns the type of the item.
Padding	Specifies the padding (space between the menu border and the item content) to display the item.
Parent	Gets the item's parent, if the current item belongs to a submenu/popup.
Position	Specifies the position of the item, within its collection.
Radio	Retrieves or sets a value that indicates whether the item is of radio type.
RadioGroup	Indicates the group of radio items that the current item belongs.
SelBackColor	Specifies the background color of the item when it is selected.
SelHotBackColor	Specifies the background color of the selected item when the cursor hovers it.
Shortcut	Specifies the key combination that the user can press to select the item quickly.
ShowAsButton	Specifies whether the item is shown as a button.
ShowAsDisabled	Specifies whether the item is shown as disabled.
ShowCheckedAsSelected	Specifies whether the checked item shows as selected.
ShowDown	Retrieves or sets a value that indicates whether the item's submenu is up or down oriented .

ShowLocalPopup	Specifies whether the item's popup is shown as local. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus.
ShowPopupArrow	Gets or sets a value that indicates whether an item that has a sub-menu shows or hides its popup arrow.
ShowPopupOnChecked	Specifies whether the item's sub menu is shown only if the item is checked.
Strikeout	Specifies whether the item's caption should appear in strikeout.
SubControl	Retrieves the Control object that holds information about item's inside component.
SubMenu	Retrieves an Items collection that indicates the item's sub menu. Retrieves Nothing, if the item contains no sub menu.
Tab	Specifies the identifier of the item/tab where the current group popup is shown instead.
Tooltip	Specifies the item's tooltip.
TooltipTitle	Specifies the title of the item's tooltip.
ToString	Loads or saves the item using string representation.
Underline	Specifies whether the item's caption appears as underlined.
UserData	Associates an extra data to the object.
Visible	Specifies whether the item is visible or hidden.

property Item.Alignment as AlignmentEnum

Retrieves or sets the item's caption alignment.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the item's

The Alignment property specifies the item's alignment. The Caption property supports built-in HTML format, so you can use the <c> to centers the item's caption or <r> to align to the right the item's caption.

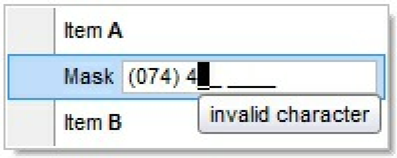
property Item.AllowEdit as AllowEditEnum

Retrieves or sets a value indicating whether the item contains an edit control.

Type	Description
AllowEditEnum	An AllowEditEnum expression that specifies whether the item displays an Edit field inside.

By default, the AllowEdit property is False, which indicates that the item displays no Edit field inside. Use the AllowEdit property to add a text-box inside the item, so the user can type any characters inside. The EditCaption property specifies the caption to be shown on the item's Edit text box. The EditWidth property specifies the width of the text-box inside the item. The EditBorder property specifies the border to be shown around the item's text box. You can use the Get method to collect all items of Edit type. The EditChange event notifies your application once the user alters the item's text-box caption. The EditOption property specifies different options to be used for a specified edit field. The ShowAsButton property specifies the whether the current item displays a button or a select button (drop down).

The following screen shot shows an item with a masking editor:



Item A

Mask (074) 4_ ____

Item B

invalid character

property Item.BackColor as Color

Specifies the item's background color of the item.

Type	Description
Color	A Color expression that specifies the item's background color. The last 7 bits in the high significant byte of the color 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 BackColor property specifies a different background color or a visual appearance for the item. The Caption property indicates the item's caption to be shown on the item. You can use the <bgcolor> HTML tag in the Caption property to specify a different background color for parts of the caption. The ForeColor property specifies the item's foreground color. The SelBackColor property specifies the item's background color when it is selected or highlighted. The HotBackColor property specifies a different background color or a visual appearance for the item, when the cursor hovers it. The SelHotBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked, and the cursor hovers it. The SelBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked.

property Item.Bold as Boolean

Specifies whether the item's caption should appear in bold.

Type	Description
Boolean	A Boolean expression that specifies whether the item's caption is shown in bold.

By default, the Bold property is False. Use the Bold property to show the item's caption in bold. The Caption property indicates the HTML caption to be shown on the item. The <b> HTML tag can be used on the item's Caption property to specify different parts of the caption to be shown in bold.

property Item.Caption as String

Retrieves or sets a value that indicates the item's caption.

Type	Description
String	A String expression that specifies the HTML caption to be displayed on the context menu.

Use the Caption property to specify the item's caption. Use the UserData property to associate any extra data to your items. Use the Tooltip property to specify the item's tooltip which can be shown when the cursor hovers the item. Use the Check property to assign a check-box to the item. Use the Radio property to assign a radio-button to the item. The ForeColor property of the Item object specifies a different foreground color for the entire item. The BackColor property of the Item object specifies a different background color / visual appearance for the entire item. The Item property searches recursively the item with giving identifier/caption. The AllowEdit property assigns an editor to an item. The CaptionWidth property specifies the fixed width to display the item's caption.

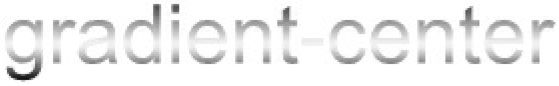
The Caption property supports the following HTML tags:

- **<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>

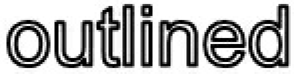
... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<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 $_{\text{subscript}}$ The "Text with <font ;7><**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The <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

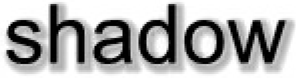
color from the current text color to gray (808080). For instance the "<font ;18><**gra** FFFFFF;1;1>gradient-center</**gra**></font>" generates the following picture:



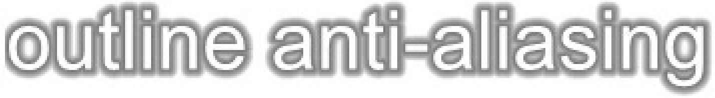
- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text.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



property Item.CaptionWidth as Long

Specifies the fixed width to display the item's caption.

Type	Description
Long	A Long expression that specifies the width to display the item's caption.

By default, the CaptionWidth property is -1. If the CaptionWidth is negative, the caption expands the size of the item to fit entirely. If the CaptionWidth property is positive, it indicates the width to display the item's caption. The Caption property specifies the HTML caption to be displayed on the item. For instance, you can use the CaptionWidth to align editors of the items. The ItemHeight property specifies the height to show the item.

property Item.Check as Boolean

Retrieves or sets a value that indicates whether the item is of check type.

Type	Description
Boolean	A Boolean expression that specifies whether the item displays a check box.

The Check property indicates whether the current item displays a check box. The Checked property specifies whether the item is checked or un-checked. The Radio property specifies whether the item displays a radio-button. The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked inside. The GetChecked property gets a collection of checked items. The GetUnchecked property gets a collection of checked items. The GetRadio method gets a safe array with the radio-items being checked within a radio group. Use the Background(exCheckBoxState0)/Background(exCheckBoxState1) property to specify the visual appearance of the check-boxes in the control. Use the UseVisualTheme property to specify whether the visual appearance for the check-boxes to be as indicated by the current XP theme. Use the ShowCheckedAsSelected property on True, to show the checked items as selected.

property Item.Checked as Boolean

Retrieves or sets a value that indicates the item's state.

Type	Description
Boolean	A Boolean expression that indicates whether the item is checked or unchecked.

The Checked property specifies whether the item is checked or un-checked. The Check property indicates whether the current item displays a check box. The Radio property specifies whether the item displays a radio-button. The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked inside. The GetChecked property gets a collection of checked items. The GetUnchecked property gets a collection of checked items. The GetRadio method gets a safe array with the radio-items being checked within a radio group. The AllowToggleRadio property on True, allows a radio button to set on zero (unchecked), if the user clicks twice the radio button. Usually, clicking a radio-button makes the previously checked radio-button in the same group, to be un-checked, and the newly clicked item to be checked. Now, if the AllowToggleRadio property is True, clicking again the radio-button, allows the radio-button to be un-checked, so allows a radio group to have no radio button checked. Use the ShowCheckedAsSelected property on True, to show the checked items as selected.

property Item.CloseOnClick as CloseOnClickEnum

Specifies the way the owner menu is closed once the user clicks the item.

Type	Description
CloseOnClickEnum	A CloseOnClickEnum expression that determines the way the hosting menu is closed when the user clicks the item.

By default, the CloseOnClick property is inherited by the control's CloseOnClick property. You can specify a different way of closing the current item by specifying a different value for Item's CloseOnClick property. Setting the CloseOnClick property on a negative value, resets that Item's CloseOnClick property, so closing the item when clicking the items is specified by the control's CloseOnClick property. For instance, you can specify the Item's CloseOnClick property on exCloseOnClickOutside, for items of button type, so the hosting menu won't be closed when the user clicks the button, and so the user can click multiple times the item without closing the menu. The ShowAsButton property indicates whether the item should look as a button. The ShowLocalPopup property specifies whether the item's popup is shown as local. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus.

property Item.Cursor as Variant

Specifies the shape of the cursor when mouse hovers the item.

Type	Description
Variant	A String expression that defines the cursor to be shown when the cursor hovers the item. The Valid values are listed bellow. Also the Cursor property could point to a cursor file to be loaded and shown while the cursor hovers the item.

By default, the Cursor property is "exDefault". Use the Cursor property of the Item object to specify a different cursor when it hovers the item only. Use the Cursor property to specify a different cursor when it hovers the menu control.

The supported values are:

- "exDefault", Standard arrow
- "exArrow", Standard arrow
- "exCross", Crosshair
- "exIBeam", I-beam
- "exIcon", Reserved
- "exSize", Reserved, use the "exSizeAll"
- "exSizeNESW", Double-pointed arrow pointing northeast and southwest
- "exSizeNS", Double-pointed arrow pointing north and south
- "exSizeNWSE", Double-pointed arrow pointing northwest and southeast
- "exSizeWE", Double-pointed arrow pointing west and east
- "exUpArrow", Vertical arrow
- "exHourglass", Hourglass
- "exNoDrop", Slashed circle
- "exArrowHourglass"
- "exHelp", Arrow and question mark
- "exSizeAll", Four-pointed arrow pointing north, south, east, and west
- "exHand", Hand

Any other value indicates the path to a cursor file to be displayed when the pointer hovers the menu control/item.

property Item.EditBorder as EditBorderEnum

Specifies the border for the inside edit control.

Type	Description
EditBorderEnum	An EditBorderEnum expression that specifies the border to be shown around the item's text box.

The EditBorder property specifies the border to be shown around the item's text box. The EditCaption property specifies the caption to be shown on the item's Edit text box. Use the AllowEdit property to add a text-box inside the item, so the user can type any characters inside. The EditWidth property specifies the width of the text-box inside the item. You can use the Get method to collect all items of Edit type. The EditChange event notifies your application once the user alters the item's text-box caption.

property Item.EditCaption as String

Specifies the edit's caption when the item contains an edit control.

Type	Description
String	A String expression that specifies the caption to be shown on the item's text box.

The EditCaption property specifies the caption to be shown on the item's Edit text box. Use the AllowEdit property to add a text-box inside the item, so the user can type any characters inside. The EditWidth property specifies the width of the text-box inside the item. The EditBorder property specifies the border to be shown around the item's text box. You can use the Get method to collect all items of Edit type. The EditChange event notifies your application once the user alters the item's text-box caption. The EditValue property indicates the edit's value.

property Item.EditMask as String

Specifies the edit's mask when the item contains an masked edit control.

Type	Description
String	A string expression that indicates the mask of the edit's field.

By default, the EditMask property is "" (empty string, no masking). The EditMask property is valid for exItemEditMask editors. The AllowEdit property associates an editor to the current item. The EditMask property specifies the mask of the edit field. The EditValue property specifies the value of the edit field, without the masking characters. The EditOption(exEditMaskFloat) specifies whether the edit field mask a floating/decimal/integer point number. The EditMask property depends on the EditOption(exEditMaskFloat) value, as explained bellow.

A) If the EditOption(exEditMaskFloat) property is False (by default), the EditMask is defined such as:

For instance, the following input-mask (ext-phone)

!(999) 000 0000;1;;select=1,empty,overtype,warning=invalid character,invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field."

indicates the following:

- The pattern should contain 3 optional digits *999*, and 7 required digits *000 0000*, aligned to the right, *!*.
- The second part of the input mask indicates *1*, which means that all literals are included when the user leaves the field.
- The entire field is selected when it receives the focus, *select=1*
- The field supports *empty* value, so the user can leave the field with no content
- The field enters in *overtype* mode, and insert-type mode is not allowed when user pressed the Insert key
- If the user enters any invalid character, a *warning* tooltip with the message "*invalid character*" is displayed.
- If the user tries to leave the field, while the field is not validated (all 7 required digits completed), the *invalid* tooltip is shown with the message "*The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field.*" The *<%mask%>* is replaced with the first part of the input mask *!(999) 000 0000*

The four parts of an input mask, or the Mask property supports up to four parts, separated by a semicolon (;). For instance, "`Time: `00:00:00;;0;overtype,warning=<fgcolor FF0000>invalid character,beep", indicates the pattern "00:00" with the prefix Time:, the masking character being the 0, instead _, the field enters in over-type mode, insert-type mode is not allowed, and the field beeps and displays a tooltip in red with the message invalid character when the user enters an invalid character.

Input masks are made up one mandatory part and three optional parts, and each part is separated by a semicolon (;). If a part should use the semicolon (;) it must uses the \; instead

The purpose of each part is as follows:

1. The first part (pattern) is mandatory. It includes the mask characters or string (series of characters) along with placeholders and literal data such as, parentheses, periods, and hyphens.

   The following table lists the placeholder and literal characters for an input mask and explains how it controls data entry:

   - **_#_**_, a digit, +, - or space (entry not required)._
   - **_0_**_, a digit (0 through 9, entry required; plus [+] and minus [-] signs not allowed)._
   - **_9_**_, a digit or space (entry not required; plus and minus signs not allowed)._
   - **_x_**_, a lower case hexa character, [0-9],[a-f] (entry required)_
   - **_X_**_, an upper case hexa character, [0-9],[A-F] (entry required)_
   - **_A_**_, any letter, digit (entry required)._
   - **_a_**_, any letter, digit or space (entry optional)._
   - **_L_**_, any letter (entry require)._
   - **_?_**_, any letter or space (entry optional)._
   - **_&_**_, any character or a space (entry required)._
   - **_C_**_, any character or a space (entry optional)._
   - **_>_**_, any letter, converted to uppercase (entry required)._
   - **_<_**_, any letter, converted to lowercase (entry required)._
   - **_*_**_, any characters combinations_
   - **_{ min,max }_** _(Range), indicates a number range. The syntax {min,max} (Range), masks a number in the giving range. The min and max values should be positive integers. For instance the mask {0,255} masks any number between 0 and 255._
   - **_[...]_** _(Alternative), masks any characters that are contained in the [] brackets. For instance, the [abcdA-D] mask any character: a,b,c,d,A,B,C,D_
   - **__**_, indicates the escape character_
   - **_t'_**_, (ALT + 175) causes the characters that follow to be converted to uppercase, until Ť(ALT + 174) is found._

    - ***Ť**, (ALT + 174) causes the characters that follow to be converted to lowercase, until ť(ALT + 175) is found.*
    - ***!**, causes the input mask to fill from right to left instead of from left to right.*

   Characters enclosed in double quotation ("" or ``) marks will be displayed literally. If this part should display/use the semicolon (;) character is should be included between double quotation ("" or ``) characters or as \; (escape).

2. The second part is optional and refers to the embedded mask characters and how they are stored within the field. If the second part is set to 0 (default, exClipModeLiteralsNone), all characters are stored with the data, and if it is set to 1 (exClipModeLiteralsInclude), the literals are stored, not including the masking/placeholder characters, if 2 (exClipModeLiteralsExclude), just typed characters are stored, if 3(exClipModeLiteralsEscape), optional, required, editable and escaped entities are included. No double quoted text is included.
3. The third part of the input mask is also optional and indicates a single character or space that is used as a placeholder. By default, the field uses the underscore (_). If you want to use another character, enter it in the third part of your mask. Only the first character is considered. If this part should display/use the semicolon (;) character is should be \; (escape)
4. The forth part of the input, indicates a list of options that can be applied to input mask, separated by comma(,) character.

   The known options for the forth part are:

    - ***float**, indicates that the field is edited as a decimal number, integer. The first part of the input mask specifies the pattern to be used for grouping and decimal separators, and - if negative numbers are supported. If the first part is empty, the float is formatted as indicated by current regional settings. For instance, "##;;;float" specifies a 2 digit number in float format. The grouping, decimal, negative and digits options are valid if the float option is present.*

    - ***grouping**=value, Character used to separate groups of digits to the left of the decimal. Valid only if float is present. For instance ";;;float,grouping=" indicates that no grouping is applied to the decimal number (LOCALE_STHOUSAND)*
    - ***decimal**=value, Character used for the decimal separator. Valid only if float is present. For instance ";;;float,grouping= ,decimal=\," indicates that the decimal number uses the space for grouping digits to the left, while for decimal separator the comma character is used (LOCALE_SDECIMAL)*

- ***negative****=value, indicates whether the decimal number supports negative numbers. The value should be 0 or 1. 1 means negative numbers are allowed. Else 0 or missing, the negative numbers are not accepted. Valid only if float is present.*
- ***digits****=value, indicates the max number of fractional digits placed after the decimal separator. Valid only if float is present. For instance, ";;;float,digits=4" indicates a max 4 digits after decimal separator (LOCALE_IDIGITS)*
- ***password****[=value], displays a black circle for any shown character. For instance, ";;;password", specifies that the field to be displayed as a password. If the value parameter is present, the first character in the value indicates the password character to be used. By default, the * password character is used for non-TrueType fonts, else the black circle character is used. For instance, ";;;password=*", specifies that the field to be displayed as a password, and use the * for password character. If the value parameter is missing, the default password character is used.*
- ***right****, aligns the characters to the right. For instance, "(999) 999-9999;;;right" displays and masks a telephone number aligned to the right.* ***readonly****, the editor is locked, user can not update the content, the caret is available, so user can copy the text, excepts the password fields.*
- ***inserttype****, indicates that the field enters in insert-type mode, if this is the first option found. If the forth part includes also the overtype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;inserttype,overtype", indicates that the field enter in insert-type mode, and over-type mode is allowed. The "##:##;;0;inserttype", indicates that the field enter in insert-type mode, and over-type mode is not allowed.*
- ***overtype****, indicates that the field enters in over-type mode, if this is the first option found. If the forth part includes also the inserttype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;overtype,inserttype", indicates that the field enter in over-type mode, and insert-type mode is allowed. The "##:##;;0;overtype", indicates that the field enter in over-type mode, and insert-type mode is not allowed.*
- ***nocontext****, indicates that the field provides no context menu when user right clicks the field. For instance, ";;;password,nocontext" displays a password field, where the user can not invoke the default context menu, usually when a right click occurs.*
- ***beep****, indicates whether a beep is played once the user enters an invalid character. For instance, "00:00;;;beep" plays a beep once the user types in invalid character, in this case any character that's not a digit.*
- ***warning****=value, indicates the html message to be shown when the user enters an invalid character. For instance, "00:00:00;;;warning=invalid character" displays a "invalid character" tooltip once the user types in invalid character, in*

this case any character that's not a digit. The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape)

- ***invalid****=value, indicates the html message to be displayed when the user enters an inappropriate value for the field. If the value is missing or empty, the option has no effect, so no validation is performed. If the value is a not-empty value, the validation is performed. If the value is single space, no message is displayed and the field is keep opened while the value is inappropriate. For instance, "!(999) 000 0000;;;invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field." displays the "The value you entered isn't appropriate for the input mask '...' specified for this field." tooltip once the user leaves the field and it is not-valid (for instance, the field includes entities required and uncompleted). The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape). This option can be combined with empty, validateas.*
- ***validateas****=value, specifies the additional validation is done for the current field. If value is missing or 0 (exValidateAsNone), the option has no effect. The validateas option has effect only if the invalid option specifies a not-empty value. Currently, the value can be 1 (exValidateAsDate), which indicates that the field is validated as a date. For instance, having the mask "!00/00/0000;;0;empty,validateas=1,invalid=Invalid date!,warning=Invalid character!,select=4,overtype", indicates that the field is validate as date (validateas=1).*
- ***empty****, indicates whether the field supports empty values. This option can be used with invalid flag, which indicates that the user can leave the field if it is empty. If empty flag is present, the field displays nothing if no entity is completed (empty). Once the user starts typing characters the current mask is displayed. For instance, having the mask "!(999) 000 0000;;;empty,select=4,overtype,invalid=invalid phone number,beep", it specifies an empty or valid phone to be entered.*
- ***select****=value, indicates what to select from the field when it got the focus. The value could be 0 (nothing, exSelectNoGotFocus), 1 (select all, exSelectAllGotFocus), 2 (select the first empty and editable entity of the field, exSelectEditableGotFocus), 3 (moves the cursor to the beginning of the first empty and editable entity of the field, exMoveEditableGotFocus), 4 (select the first empty, required and editable entity of the field, exSelectRequiredEditableGotFocus), 5 (moves the cursor to the beginning of the first empty, required and editable entity of the field,*

exMoveRequiredEditableGotFocus). For modes 2 and 4 the entire field is selected if no matching entity is found. For instance, "`Time:`XX:XX;;;select=1" indicates that the entire field (including the Time: prefix) is selected once it get the focus. The "`Time:`XX:XX;;;select=3", moves the cursor to first X, if empty, the second if empty, and so on

Experimental:
***multiline**, specifies that the field supports multiple lines.*
***rich**, specifies that the field displays a rich type editor. By default, the standard edit field is shown*
***disabled**, shows as disabled the field.*

B) If the EditOption(exEditMaskFloat) property is True, the EditMask is defined such as:

The EditMask property may indicate the followings:

- **negative number**: if the first character in the mask is - (minus) the control supports negative numbers. Pressing the - key will toggle the sign of the number. The + sign is never displayed.
- **decimal symbol**: the last character that's different than # (digit), or 0 (zero) indicates the decimal symbol. If it is not present the control mask a floating point number without decimals.
- **thousand symbol**: the thousand symbol is the last character that's not a # (digit), 0 (zero) or it is not the decimal symbol as explained earlier, if present.
- the maximum **number of decimals** in the number (the # or 0 character after the decimal symbol)
- the maximum number of digits in the integer part (the number of # or 0 character before decimal symbol)
- the **0** character indicates **a leading-zero**. The count of 0 (zero) characters before decimal character indicates the leading-zero for integer part of the control, while the count of 0 (zero) characters after the decimal separator indicates the leading-zero for decimal part of the control. For instance, the Mask on "-###,###,##0.00", while the control's Text property is 1, the control displays 1.00, if 1.1 if displays 1.10, and if empty, the 0.00 is displayed.

If the EditMask property is empty, the control takes the settings for the regional options like: Decimal Symbol , No. of digits after decimal, Digit grouping symbol.

Here are few samples:

The EditMask"**-###.###.##0,00**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator). This format displays leading-zeros.

The EditMask"**-###.###.###,##**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator)

The EditMask"**-###,###,###.##**" filter floating point numbers a number for English settings ("." is the decimal sign, "," is the thousands separator)

The EditMask"**####**" indicates a max-4 digit number (positive) without a decimal symbol and without digit grouping

The EditMask"**-##.#**" filters a floating point number from the -99.9 to 99.9 ("." is the decimal sign, no thousands separator)

The EditMask"**#,###.##**" filters a floating point number from the 0 to 9,999.99 with digit grouping ("." is the decimal sign, "," is the thousands separator).

property Item.EditOption(Option as EditOptionEnum) as Variant

Specifies different options for item's edit control.

Type	Description
Option as EditOptionEnum	An EditOptionEnum expression that specifies the option to be updated.
Variant	A VARIANT expression that indicates the option's value.

The EditOption property different options for item's edit control. The AllowEdit property associates an editor to the current item. The EditCaption property specifies the value to show in the edit field. The EditWidth property specifies the size/width of the edit field inside the item. The EditBorder property specifies whether the edit shows a border around it. The control fires the EditChange event when the user changes the edit's caption. For instance, the EditOption(exEditSpinStep) property specifies the step to advance when user clicks the editor's spin.

property Item.EditValue as Variant

Specifies the edit's value when the item contains an edit control.

Type	Description
Variant	A VARIANT expression that specifies the edit's value.

The EditValue/EditCaption property specifies the caption to be shown on the item's edit text box. Use the AllowEdit property to add a text-box inside the item, so the user can type any characters inside. The EditWidth property specifies the width of the text-box inside the item. The EditBorder property specifies the border to be shown around the item's text box. You can use the Get method to collect all items of Edit type. The EditChange event notifies your application once the user alters the item's text-box caption.

The EditValue property indicates the edit's value as shown bellow:

- The EditValue property specifies the value of the edit field (string expression), without the masking characters, when AllowEdit property includes the exItemEditMask flag.
- The EditValue property indicates the current slider position/value (long expression), when AllowEdit property includes the exItemEditSlider, exItemEditProgress, exItemEditScrollBar, or exItemEditColor flag.

property Item.EditWidth as Long

Specifies the width for the inside edit control.

Type	Description
Long	A Long expression that specifies the width of the item's text box.

The EditWidth property specifies the width of the text-box inside the item. The EditBorder property specifies the border to be shown around the item's text box. The EditCaption property specifies the caption to be shown on the item's Edit text box. Use the AllowEdit property to add a text-box inside the item, so the user can type any characters inside. You can use the Get method to collect all items of Edit type. The EditChange event notifies your application once the user alters the item's text-box caption.

property Item.Enabled as Boolean

Retrieves or sets a value that indicates whether 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. Use the Enabled property to disable an item. A disabled item (Enabled property is False) shows as grayed, and it is un-selectable, so the user can select or highlight it. An item (Enabled property is True, ShowAsDisabled property is True), shows as grayed, but it is selectable, so the user can select or highlight it. You can use the Visible property to show or hide the item. The Remove method removes an individual Item object giving its identifier or caption. Use the ShowAsDisabled property to show the item as disabled, while it is enabled.

property Item.ForeColor as Color

Specifies the item's foreground color of the item.

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

The ForeColor property specifies the item's foreground color. The BackColor property specifies a different background color or a visual appearance for the item. The Caption property indicates the item's caption to be shown on the item. You can use the <fgcolor> HTML tag in the Caption property to specify a different foreground color for parts of the caption. The SelForeColor property specifies the item's foreground color when it is selected or highlighted.

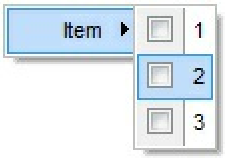
property Item.GroupPopup as GroupPopupEnum

Specifies whether the items of the sub-menu are grouped and displayed on the current item.

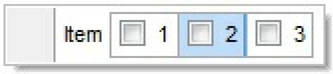
Type	Description
GroupPopupEnum	A GroupPopupEnum expression that specifies whether the items of the sub-menu are grouped and displayed on the current item.

By default, the GroupPopup property is exNoGroupPopup, which indicates that the sub-menu is not shown as grouped. Use the GroupPopup property to show the item's sub-menu as a group in the current item. The SubMenu property indicates the item's sub-menu. The GroupPopup has no effect if the item has no sub-items.

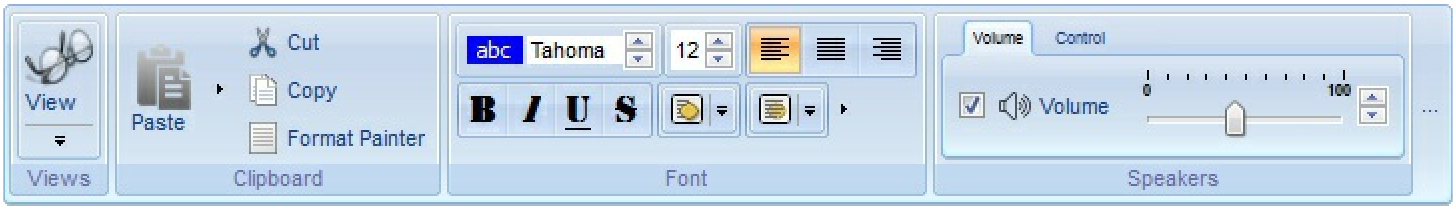
The following screen shot shows the items with no grouping:



The following screen shot shows the items with grouping (horizontally):



The following screen shot shows different type of items grouped horizontally, vertically:



property Item.HotBackColor as Color

Specifies the hot background color of the item (when the cursor hovers the item).

Type	Description
Color	A Color expression that specifies the item's background color, when the cursor hovers it. The last 7 bits in the high significant byte of the color 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 HotBackColor property specifies a different background color or a visual appearance for the item, when the cursor hovers it. The BackColor property specifies the item's background color of the item. The SelBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked. The Caption property indicates the item's caption to be shown on the item. You can use the <bgcolor> HTML tag in the Caption property to specify a different background color for parts of the caption. The SelHotBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked, and the cursor hovers it.

property Item.HTMLImage as String

Retrieves or sets a value that indicates the key of the image (HTMLPicture method) to be displayed on the item (left side).

Type	Description
String	A String expression that indicates the key of the picture to be displayed on the left side of the caption.

The HTMLImage property assigns a picture to the left side of the caption. The key of the picture to be displayed must be loaded previously using the HTMLPicture property. The HTMLImage property has effect only if the Image property is -1 (by default). Use the Image property to assign an icon from the Images collection to the left side of the caption. Use the FlatImageWidth property to specify the width of the column that displays icons/images/check or radio buttons.

The following VFP samples loads the picture using the HTMLPicture method, and displays it on the left side of the caption using the HTMLImage property.

```
contextMenu = CreateObject("Exontrol.ContextMenu")
with contextMenu
.HTMLPicture("pic1") = "C:\exontrol\images\colorize.gif"
.Items.ToString = "Item A[himg=pic1]"
iSelect = .Select()
IF ( iSelect # 0 ) then
?( .Items.item(iSelect).Caption )
ENDIF
endwith
```

or:

```
contextMenu = CreateObject("Exontrol.ContextMenu")
with contextMenu
.HTMLPicture("pic1") = "C:\exontrol\images\colorize.gif"
.Items.Add("Item A").HTMLImage = "pic1"
iSelect = .Select()
IF ( iSelect # 0 ) then
?( .Items.item(iSelect).Caption )
ENDIF
endwith
```

These two samples are equivalent.

property Item.ID as Long

Retrieves or sets a value that specifies the item's identifier.

Type	Description
Long	A Long expression that defines the item's identifier. This property can be specified at adding time, by using the ID parameter of the Add method.

Use the ID property to associate an unique identifier to each item. You can use the Item property of the exContextMenu component to get the Item object based on its identifier. Use the Debug property to display the identifiers for all visible items, for debugging purposes. The First number in the [] parenthesis indicates the item's ID property. Use the Caption property to specify the item's caption. Use the UserData property to associate any extra data to your items.

property Item.Image as Long

Retrieves or sets a value that indicates the item's index image.

Type	Description
Long	A Long expression that specifies the zero-based index of the icon to be displayed on the item.

By default, the Image property is -1, which indicates no icon associated. The Image is displayed on the left side of the item's caption. The Images method should be used to specify a collection of icons to be used by the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The <img> tag can be used in the Caption property of the Item object to display an Icon or a custom-size picture. Use the HTMLImage property to assign a BMP, JPG, GIF or PNG file to left side of the caption, the same way as you will do with the Image property.

property Item.Italic as Boolean

Specifies whether the item's caption should appear in italic.

Type	Description
Boolean	A Boolean expression that specifies whether the item's caption is shown in italic.

By default, the Italic property is False. Use the Italic property to show the item's caption in italic. The Caption property indicates the HTML caption to be shown on the item. The <i> HTML tag can be used on the item's Caption property to specify different parts of the caption to be shown in italic.

property Item.ItemHeight as Long

Specifies the fixed height to display the item.

Type	Description
Long	A Long expression that specifies the height of the item

The ItemHeight property specifies the height to display the item. By default, the ItemHeight property is -1, which indicates that the control sets the item height based on on its content. If the ItemHeight property is positive, it indicates the height to display the item. The CaptionWidth property specifies the fixed width to display the item's caption.

property Item.Items as Items

Retrieves an Items collection that indicates the item's sub menu. Retrieves Nothing, if the item contains no sub menu.

Type	Description
Items	An Items collection that holds the Item objects to be displayed in the sub-menu.

The Items and SubMenu properties are equivalents. Use the Items property to access the Item objects in a sub-menu item. The Parent property of the Item object returns an empty object, if the item contains no parent. The Parent item property can be used to access the parent of the item, when it is contained by a sub-menu.

property Item.ItemType as ItemTypeEnum

Returns the type of the item.

Type	Description
ItemTypeEnum	An ItemTypeEnum expression that specifies the type of the item.

The ItemType property is a read-only property that gets the type of the item. Use the Debug property to display debugging information in the item's Caption. Use the Get method to get a collection of Item objects that meet your criteria.

property Item.Padding as String

Specifies the padding (space between the menu border and the item content) to display the item.

Type	Description
String	A string expression that indicates a list of 4 positive numbers separated by comma characters, which indicates the distance in pixels from margin to client, in the following format: left, top, right, bottom.

By default, the Padding property is empty (0,0,0,0). The Padding property specifies the padding for a particular item. The Padding property specifies the padding (space between the menu border and the item content) to display the item. The Caption property indicates the item's caption to be shown on the item. The BackColor property specifies a different background color or a visual appearance for the item.

property Item.Parent as Item

Gets the item's parent, if the current item belongs to a submenu/popup.

Type	Description
Item	An Item object that specifies the parent item of the current item.

The Parent property of the Item object returns an empty object, if the item contains no parent. The Parent item property can be used to access the parent of the item, when it is contained by a sub-menu. Use the Items property to access the Item objects in a sub-menu item.

property Item.Position as Long

Specifies the position of the item, within its collection.

Type	Description
Long	A Long expression that specifies the position of the item.

The Position property specifies the position of the item, within its collection.

property Item.Radio as Boolean

Retrieves or sets a value that indicates whether the item is of radio type.

Type	Description
Boolean	A Boolean expression that specifies whether the item displays a radio button.

The Radio property specifies whether the item displays a radio-button. The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked. The Checked property specifies whether the item is checked or un-checked. The GetRadio method gets a safe array with the radio-items being checked within a radio group. Use the Background(exRadioButtonState0)/Background(exRadioButtonState1) property to specify the visual appearance of the radio-buttons in the control. Use the UseVisualTheme property to specify whether the visual appearance for the radio-buttons to be as indicated by the current XP theme. The AllowToggleRadio property on True, allows a radio button to set on zero (unchecked), if the user clicks twice the radio button. Usually, clicking a radio-button makes the previously checked radio-button in the same group, to be un-checked, and the newly clicked item to be checked. Now, if the AllowToggleRadio property is True, clicking again the radio-button, allows the radio-button to be un-checked, so allows a radio group to have no radio button checked. Use the ShowCheckedAsSelected property on True, to show the checked items as selected.

property Item.RadioGroup as Long

Indicates the group of radio items that the current item belongs.

Type	Description
Long	A Long expression that specifies the identifier of the radio group. A radio group allows a single radio-item to be checked inside. If the RadioGroup property is not specified, all radio-buttons in the controls are in the same group 0.

The RadioGroup property specifies a group of radio-buttons. A radio group allows a single radio-item to be checked inside. The Radio property specifies whether the item displays a radio-button. The Checked property specifies whether the item is checked or un-checked. The GetRadio method gets a safe array with the radio-items being checked within a radio group. Use the Background(exRadioButtonState0)/Background(exRadioButtonState1) property to specify the visual appearance of the radio-buttons in the control. Use the UseVisualTheme property to specify whether the visual appearance for the radio-buttons to be as indicated by the current XP theme. The AllowToggleRadio property on True, allows a radio button to set on zero (unchecked), if the user clicks twice the radio button. Usually, clicking a radio-button makes the previously checked radio-button in the same group, to be un-checked, and the newly clicked item to be checked. Now, if the AllowToggleRadio property is True, clicking again the radio-button, allows the radio-button to be un-checked, so allows a radio group to have no radio button checked.

property Item.SelBackColor as Color

Specifies the background color of the item when it is selected.

Type	Description
Color	A Color expression that specifies the item's background color, when the item is selected/checked. The last 7 bits in the high significant byte of the color 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 SelBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked. The BackColor property specifies the item's background color of the item. The SelHotBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked, and the cursor hovers it. The HotBackColor property specifies a different background color or a visual appearance for the item, when the cursor hovers it. The Caption property indicates the item's caption to be shown on the item. You can use the <bgcolor> HTML tag in the Caption property to specify a different background color for parts of the caption.

property Item.SelHotBackColor as Color

Specifies the background color of the selected item when the cursor hovers it.

Type	Description
Color	A Color expression that specifies the item's background color, when the item is selected/checked and the cursor hovers it. The last 7 bits in the high significant byte of the color 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 SelHotBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked, and the cursor hovers it. The SelBackColor property specifies a different background color or a visual appearance for the item, when item is selected / checked. The BackColor property specifies the item's background color of the item. The HotBackColor property specifies a different background color or a visual appearance for the item, when the cursor hovers it. The Caption property indicates the item's caption to be shown on the item. You can use the <bgcolor> HTML tag in the Caption property to specify a different background color for parts of the caption.

property Item.Shortcut as String

Specifies the key combination that the user can press to select the item quickly.

Type	Description
String	A character that specifies the key combination that the user can press to select the item quickly. A null character indicates that no shortcut is associated with the item.

By default, the Shortcut property is defined as first sequence found in the item's Caption between <u> and </u> HTML tags. Pressing the shortcut key is similar with selecting the item and pressing the Enter key. The shortcuts in the context menu have effect only if the IncrementalSearch property is exNoIncrementalSearch.

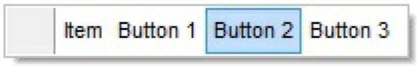
property Item.ShowAsButton as ShowAsButtonEnum

Specifies whether the item is shown as a button.

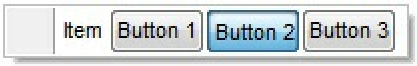
Type	Description
ShowAsButtonEnum	A ShowAsButtonEnum expression that indicates whether the item is shown as a button.

By default, the ShowAsButton property is False. Use the ShowAsButton property to add buttons to your item. The Caption property specifies the caption of the item/button. Use the Item's CloseOnClick property to specify a different way to close the menu when user clicks a specified item. You can use the ShowAsButton property on exShowAsSelectButton, for a popup-item, where the SubMenu property determines the sub-menu/items to be shown when user clicks the associated arrow (select button).

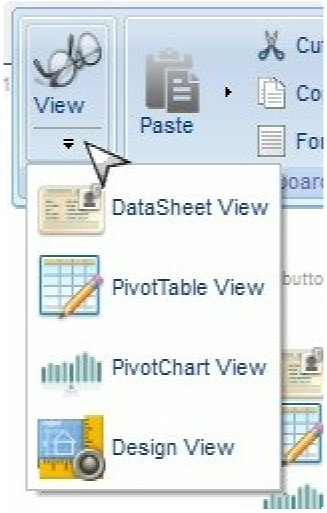
The following screen shot shows the items with no button appearance (ShowAsButton on exShowAsButtonNone)



The following screen shot shows the items with button appearance (ShowAsButton on exShowAsButton)



The following screen shot shows the items with button appearance (ShowAsButton on exShowAsSelectButtonBottom)



property Item.ShowAsDisabled as Boolean

Specifies whether the item is shown as disabled.

Type	Description
Boolean	A Boolean expression that specifies whether the current item is shown as disabled.

By default, the ShowAsDisabled property is False. Use the ShowAsDisabled property to shows the current item as disabled. The Enabled property specifies whether the item is enabled or disabled. The ShowAsDisabled property does not change the Enabled property, the item acts like an enabled item. For instance, you can not highlight a disabled item, instead you can highlight an item that looks as disabled. A disabled item (Enabled property is False) shows as grayed, and it is un-selectable, so the user can select or highlight it. An item (Enabled property is True, ShowAsDisabled property is True), shows as grayed, but it is selectable, so the user can select or highlight it.

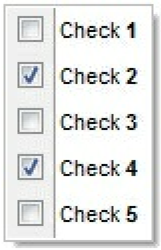
property Item.ShowCheckedAsSelected as ShowCheckedAsSelectedEnum

Specifies whether the checked item shows as selected.

Type	Description
ShowCheckedAsSelectedEnum	A ShowCheckedAsSelectedEnum expression that specifies whether the checked item shows as selected.

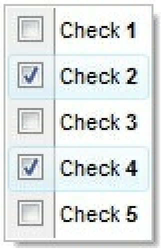
By default, the ShowCheckedAsSelected property is exDisplayItemCheckInherit, which indicates that the control's ShowCheckedAsSelected property specifies how the checked item is shown. Use the ShowCheckedAsSelected property on non zero, to show the checked items as selected. A checked item is an item with the Check or Radio property set on True and the Checked property is True. The SelBackColor property indicates the color to show background of the selected / highlighted item. The AllowToggleRadio property on True, allows a radio button to set on zero (unchecked), if the user clicks twice the radio button. The ShowCheckedAsSelectedTransparency property specifies the transparency (percent) to show the checked items when selected.

The following screen shot shows the control when the ShowCheckedAsSelected property is exDisplayItemCheckDefault(by default):



☐ Check 1
☑ Check 2
☐ Check 3
☑ Check 4
☐ Check 5

The following screen shot shows the control when the ShowCheckedAsSelected property is exDisplayItemCheckHighlight:



☐ Check 1
☑ Check 2
☐ Check 3
☑ Check 4
☐ Check 5

property Item.ShowDown as Boolean

Retrieves or sets a value that indicates whether the item's submenu is up or down oriented .

Type	Description
Boolean	A Boolean expression that specifies whether the sub-menu is shown down or up to the item

By default, the ShowDown property is True. Use the ShowDown property to show the sub-menu up to the item. The SubMenu/Items property accesses the collection of Item objects to be shown on the sub-menu.

property Item.ShowLocalPopup as Boolean

Specifies whether the item's popup is shown as local. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus.

Type	Description
Boolean	A Boolean expression that specifies whether the item's sub-menu is closed when user clicks an item.

By default, the ShowLocalPopup property is False. Use the ShowLocalPopup property on True, to provide drop down list. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus. The LocalAppearance property specifies a different visual appearance for the local popup. Use the CloseOnClick property to specify how to close the current popup when user clicks a specified item. The PopupAppearance specifies a different visual appearance for the current submenu.

property Item.ShowPopupArrow as Boolean

Gets or sets a value that indicates whether an item that has a sub-menu shows or hides its popup arrow.

Type	Description
Boolean	A Boolean expression that indicates whether an item that has a sub-menu shows or hides its popup arrow.

The ShowPopupArrow property indicates whether an item that has a sub-menu shows or hides its popup arrow.

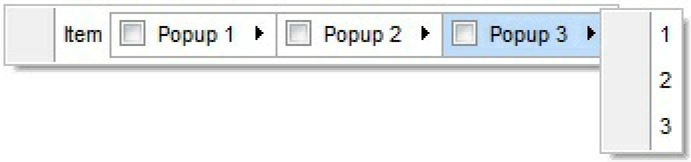
property Item.ShowPopupOnChecked as Boolean

Specifies whether the item's sub menu is shown only if the item is checked.

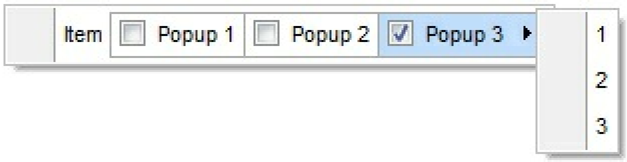
Type	Description
Boolean	A Boolean expression that specifies whether the item's sub menu is shown only if the item is checked.

By default, the ShowPopupOnChecked property is False. The Check property assigns a check box to the current item. The SubMenu property specifies the sub-items of the current item. The ShowPopupOnChecked property has effect only if the item displays sub items (the SubMenu.Count property is not zero). The ShowCheckedAsSelected property specifies how the checked item is displayed.

The following screen shot show items with ShowPopupOnChecked on False (default) *(please notice that all items display the popup-arrow)*:



The following screen shot show items with ShowPopupOnChecked on True *(please notice that just checked popup displays the popup-arrow, and so the sub-menu)*:



property Item.Strikeout as Boolean

Specifies whether the item's caption should appear in strikeout.

Type	Description
Boolean	A Boolean expression that specifies whether the item's caption is shown in strikeout.

By default, the Strikeout property is False. Use the Strikeout property to show the item's caption in strikeout. The Caption property indicates the HTML caption to be shown on the item. The <s> HTML tag can be used on the item's Caption property to specify different parts of the caption to be shown in strikeout.

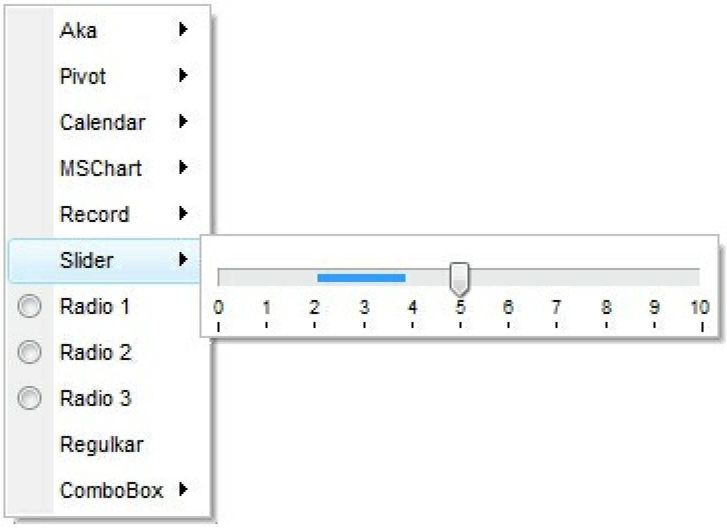
property Item.SubControl as Control

Retrieves the Control object that holds information about item's inside component.

Type	Description
Control	A Control object that holds properties to handle the inside ActiveX item.

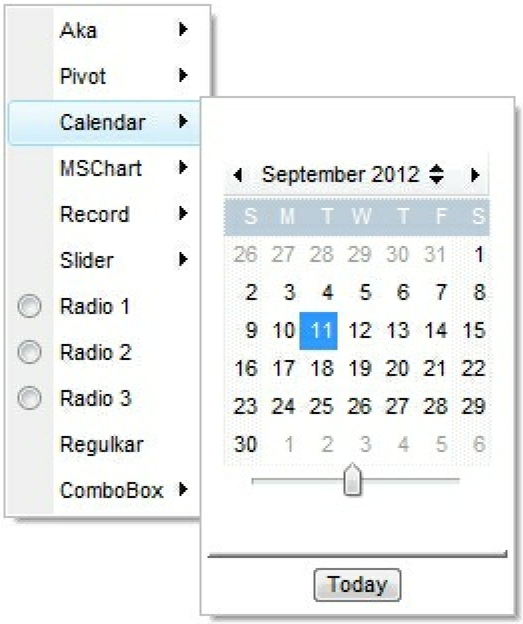
Use the SubControl property when using the ItemTypeEnum.SubControl to add an item that hosts an ActiveX inside. Use the ControlID property to specify the IDentifier of the object to be displayed on the item. Use the Create method to create an inside ActiveX control. The inside ActiveX control fires the events through the OleEvent event.

The following screen shot displays an item with an ExSlider inside:



3

The following screen shot displays an item with an ExCalendar inside:



The following samples shows how to load an ActiveX control (Exontrol.Calendar)

VB6,VBA (MS Access, Excell...),VB.NET for /COM

```
With CreateObject("Exontrol.ContextMenu")
    With .Items.Add("Calendar",3).SubControl
        .ControlID = "Exontrol.Calendar"
        .Create
    End With
    .Select
End With
```

VB.NET

```
' Add 'exontrol.excontextmenu.dll' reference to your project.
With New exontrol.EXCONTEXTMENULib.excontextmenu()
    With .Items.Add("Calendar",3).SubControl
        .ControlID = "Exontrol.Calendar"
        .Create()
    End With
    .Select()
End With
```

C++

```
/*
    Includes the definition for CreateObject function like follows:
    #include <comdef.h>
    IUnknownPtr CreateObject( BSTR Object )
    {
        IUnknownPtr spResult;
        spResult.CreateInstance( Object );
        return spResult;
    };
*/
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXCONTEXTMENULib' for the library: 'ExContextMenu 1.0 Type Library'
    #import <ExContextMenu.dll>
    using namespace EXCONTEXTMENULib;
*/
EXCONTEXTMENULib::IExContextMenuPtr var_ExContextMenu = ::CreateObject(L"Exontrol.ContextMenu");
    EXCONTEXTMENULib::IControlPtr var_Control = var_ExContextMenu->GetItems()->Add(L"Calendar",long(3),vtMissing)->GetSubControl();
        var_Control->PutControlID(L"Exontrol.Calendar");
        var_Control->Create();
    var_ExContextMenu->Select(vtMissing,vtMissing,vtMissing);
```

C++ Builder

```
/*
    Select the Component\Import Component...\Import a Type Library,
    to import the following Type Library:
        ExContextMenu 1.0 Type Library
    TypeLib: e:\Exontrol\ExContextMenu\project\Site\ExContextMenu.dll
    to define the namespace: Excontextmenulib_tlb
*/
//#include "EXCONTEXTMENULIB_TLB.h"
Excontextmenulib_tlb::IExContextMenuPtr var_ExContextMenu = Variant::CreateObject(L"Exontrol.ContextMenu");
    Excontextmenulib_tlb::IControlPtr var_Control = var_ExContextMenu->Items->Add(L"Calendar",TVariant(3),TNoParam())->SubControl;
        var_Control->ControlID = L"Exontrol.Calendar";
        var_Control->Create();
    var_ExContextMenu->Select(TNoParam(),TNoParam(),TNoParam());
```

C#

```
// Add 'exontrol.excontextmenu.dll' reference to your project.
exontrol.EXCONTEXTMENULib.excontextmenu var_ExContextMenu = new exontrol.EXCONTEXTMENULib.excontextmenu();
    exontrol.EXCONTEXTMENULib.Control var_Control = var_ExContextMenu.Items.Add("Calendar",3,null).SubControl;
        var_Control.ControlID = "Exontrol.Calendar";
        var_Control.Create();
```

```
   var_ExContextMenu.Select(null,null,null);
```

C# for /COM

```
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
EXCONTEXTMENULib.ExContextMenu var_ExContextMenu = new
EXCONTEXTMENULib.ExContextMenu();
   EXCONTEXTMENULib.Control var_Control =
var_ExContextMenu.Items.Add("Calendar",3,null).SubControl;
      var_Control.ControlID = "Exontrol.Calendar";
      var_Control.Create();
   var_ExContextMenu.Select(null,null,null);
```

X++ (Dynamics Ax 2009)

```
COM com_Control,com_ExContextMenu,com_Items,com_item;
anytype var_Control,var_ExContextMenu,var_Items,var_item;
;
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
var_ExContextMenu = COM::createFromObject(new
EXCONTEXTMENULib.excontextmenu()); com_ExContextMenu = var_ExContextMenu;
   var_Items = COM::createFromObject(com_ExContextMenu.Items()); com_Items =
var_Items;
   var_item =
COM::createFromObject(com_Items).Add("Calendar",COMVariant::createFromInt(3));
com_item = var_item;
   var_Control = com_item.SubControl(); com_Control = var_Control;
      com_Control.ControlID("Exontrol.Calendar");
      com_Control.Create();
   com_ExContextMenu.Select();
```

Delphi 8 (.NET only)

```
with (ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMenu'))
as EXCONTEXTMENULib.ExContextMenu) do
begin
   with Items.Add('Calendar',TObject(3),Nil).SubControl do
   begin
```

```
        ControlID := 'Exontrol.Calendar';
        Create();
    end;
    Select(Nil,Nil,Nil);
end;
```

Delphi (standard)

```
with
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMen
 as EXCONTEXTMENULib_TLB.ExContextMenu) do
begin
    with Items.Add('Calendar',OleVariant(3),Null).SubControl do
    begin
        ControlID := 'Exontrol.Calendar';
        Create();
    end;
    Select(Null,Null,Null);
end;
```

VFP

```
with CreateObject("Exontrol.ContextMenu")
    with .Items.Add("Calendar",3).SubControl
        .ControlID = "Exontrol.Calendar"
        .Create
    endwith
    .Select()
endwith
```

XBasic (Alpha Five)

```
' Occurs when the user presses and then releases the left mouse button over
the control.
function Click as v ()
    Dim oPivot as P
    Dim var_Control as P
    Dim var_ExContextMenu as P
    oPivot = topparent:CONTROL_ACTIVEX1.activex
    var_ExContextMenu = OLE.Create("Exontrol.ContextMenu")
        var_Control = var_ExContextMenu.Items.Add("Calendar",3).SubControl
            var_Control.ControlID = "Exontrol.Calendar"
            var_Control.Create()
        var_ExContextMenu.Select()
end function

Dim oPivot as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
```

Visual Objects

```
local var_ExContextMenu as IExContextMenu
// Generate Source for 'ExContextMenu 1.0 Type Library' server from
Tools\Automation Server...
var_ExContextMenu := IExContextMenu{"Exontrol.ContextMenu"}
    var_Control := var_ExContextMenu:Items:Add("Calendar",3,nil):SubControl
        var_Control:ControlID := "Exontrol.Calendar"
        var_Control:Create()
    var_ExContextMenu:Select(nil,nil,nil)
```

property Item.SubMenu as Items

Retrieves an Items collection that indicates the item's sub menu. Retrieves Nothing, if the item contains no sub menu.

Type	Description
Items	An Items collection that holds the Item objects to be displayed in the sub-menu.

The Items and SubMenu properties are equivalents. Use the SubMenu property to access the Item objects in a sub-menu item. The Parent property of the Item object returns an empty object, if the item contains no parent. The Parent item property can be used to access the parent of the item, when it is contained by a sub-menu.

property Item.Tab as Long

Specifies the identifier of the item/tab where the current group popup is shown instead.

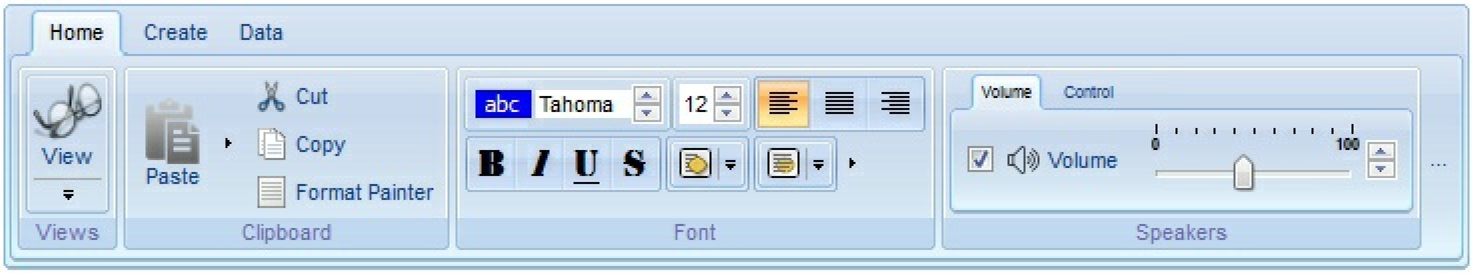
Type	Description
Long	A Long expression that specifies the identifier of the item where the grouping items of the current item is shown.

By default, the Tab property is 0. The Tab property specifies the identifier of the item where the grouping items of the current item is shown. Use the Tab property to simulate Tab/Pages into your control. By default, the grouping items are displayed right/bottom after the item. The GroupPopup property specifies the way the grouping items are shown. Using the Tab property you can specify where the current grouping items/submenu is shown. Use the BackColor/HotBackColor properties to specify the background color/visual appearance for the grouping items. Use the BackColor/HotBackColor/SelBackColor/SelHotBackColor properties to specify the background color/visual appearance of a specified item

You can use the Tab property in combination with the following properties:

- Radio/Check to specify a radio or check-type item. Usually this item indicates the page of the tab
- ShowPopupOnChecked property to specify that its current popup to be shown only if checked
- ShowCheckedAsSelected property to show a checked item as selected/highlighted instead displaying the check/radio button

The following screen shot shows the Tabbed view feature with EBN files:



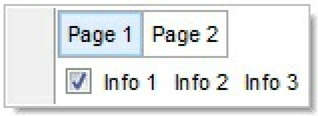
For instance, the following VB sample:

```
Dim context As New EXCONTEXTMENULib.ExContextMenu


Private Sub Form_Load()
    With context
        .Items.ToString = "[group=0x103]([group](Page 1[rad=1][group=3][spchk][tab=999]
[show=1](Info 1[chk=1],Info 2,Info 3),Page 2[rad][group=3][spchk][tab=999][show=1]
(Info 4,Info 5,Info 6)),[id=999])"
    End With
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    context.Select
End Sub
```

generates the following screen shot:



property Item.Tooltip as String

Specifies the item's tooltip.

Type	Description
String	A String expression that defines the HTML caption to be displayed when the cursor hovers the item.

The Tooltip property assigns a HTML tooltip to an item, that's displayed only when the cursor hovers the item. The TooltipTitle property specifies the title for the item's tooltip. The TooltipDelay property specifies the time until the tooltip is shown. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipFont property specifies 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 ToolTip property supports the following HTML tags:

- **<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.

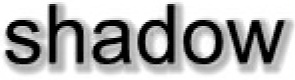
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values 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 $_{\text{subscript}}$ The "Text with <font ;7><**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The <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.TooltipTitle as String

Specifies the title of the item's tooltip.

Type	Description
String	A String expression that specifies the title of the item's tooltip.

The TooltipTitle property specifies the title for the item's tooltip. The Tooltip property assigns a HTML tooltip to an item, that's displayed only when the cursor hovers the item. The TooltipDelay property specifies the time until the tooltip is shown. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipFont property specifies 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.

property Item.ToString as String

Loads or saves the item using string representation.

Type	Description
String	A String expression that specifies the item to be loaded/saved.

The ToString property of the Items object, builds the context-menu using a string, rather than adding item one by one. The control's setup installs the WYSIWYG EditContextMenu Tool, that helps you to define the ToString format.

The ToString property looks like follows:

```
[id=10][group=0x03]([id=10][group=0x03][itemspad=4,4,4,4][itemsbghot=0x1F000000](
Annoyed1[id=20],Bunny2[id=30],[id=50][edittype=0x01][editwidth=-100],
Cellphone3[id=40]),[id=10][group=0x03][itemspad=4,4,4,4][itemsbghot=0x1F000000](
Annoyed1[id=20],Bunny2[id=30],Cellphone3[id=40]))
```

Each item is followed by its options, and its sub-items between () parentheses. The item's option includes the icons, pictures, edit attributes and so on.

The ToString syntax in BNF notation:

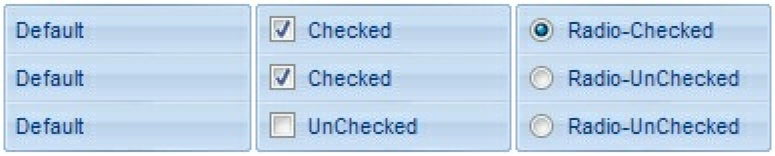
<ToString> ::= <ITEMS>
<ITEMS> ::= <ITEM>["("<ITEMS>")"][","<ITEMS>]
<ITEM> ::= <CAPTION>[<OPTIONS>]
<OPTIONS> ::= "["<OPTION>"]"["["<OPTIONS>"]"]
<OPTION> ::= <PROPERTY>["="<VALUE>]
<PROPERTY> ::= "img" | "himg" | "sep" | "id" | "typ" | "group" | "chk" | "button" | "align" | "spchk" | "show" | "rad" | "dis" | "showdis" | "bld" | "itl" | "stk" | "und" | "bg" | "fg" | "edittype" | "edit" | "mask" | "border" | "editwidth" | "captionwidth" | "height" | "grp" | "tfi" | "ttp" | "min" | "max" | "tick" | "freq" | "ticklabel" | "small" | "large" | "spin" | "ettp" | "float" | "close" | "local" | "popupapp" | "itemspad" | "itemsbg" | "itemsbghot" | "itemsbgext" | "visible" | "tab" | "pad" | "bghot" | "bgsel" | "bgselhot" | "arrow" | "popupalign" | "popupoffset" | "popupat"

where the <CAPTION> is the HTML caption to be shown on the context menu item. The <VALUE> indicates the value of giving property.

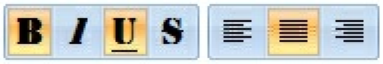
- id=<VALUE>, where <VALUE> is an integer expression, that indicates the identifier of the item.
- bg=<VALUE>, specifies the item's background color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value,

and the BB is the blue value), or an integer expression to that refers an EBN object.

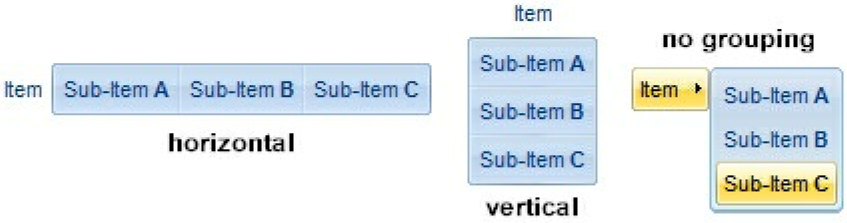
- bghot=<VALUE>, specifies the item's background color, while the cursor hovers the item, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- bgsel=<VALUE>, specifies the item's background color, while the item is checked/selected, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- bgselhot=<VALUE>, specifies the item's background color, while the item is checked/selected and the cursor hovers it, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- fg=<VALUE>, specifies the item's foreground color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or a integer expression.
- sep, specifies an separator item
- dis, specifies a disabled item
- showdis=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the item shows as disabled, but it is still enabled
- bld, specifies that the item appears in bold
- itl, specifies that the item appears in italics
- stk, specifies that the item appears as strikeout
- und, specifies that the item is underlined
- align=<VALUE>, where <VALUE> could be one of the following:
    - **0** (left), to align the item's caption to the left
    - **1** (center), to center the item's caption
    - **2** (right), to align the item's caption to the right
- captionwidth=<VALUE>, specifies the width to show the HTML caption of the item. where <VALUE> could be a integer expression. A negative value indicates that no limitation is applied to the item's caption, so no truncate caption is shown
- height=<VALUE>, specifies the height to show the item, where <VALUE> could be a positive integer expression
- pad=<VALUE>, specifies the padding (space between the menu border and the item content) to display the item. The <VALUE> is a list of coordinates such as left,top,right,bottom
- img=<VALUE>, where <VALUE> is an integer expression, that indicates the index of the icon being displayed for the item.
- himg=<VALUE>, where <VALUE> indicates the key of the picture to be displayed for the item.



- typ=<VALUE>, where <VALUE> could be one of the following:
    - **0** for default/regular items (no check/radio button is associated with the item),
    - **1** for items that display a check/box (chk),
    - **2** to display radio buttons (rad)
- chk[=<VALUE>], where <VALUE> could be **0** for unchecked, or **not zero** for checked. The chk option makes the item to display a check box. If the <VALUE> is missing the item still displays an un-checked check box.
- rad=<VALUE>, where <VALUE> could be **0** for unchecked radio button or **not zero** to for checked radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored.
- grp=<VALUE>, defines the radio group. It should be used when you define more groups of radio buttons. A group of radio buttons means that only one item could be checked at one time. The rad option specifies that the item displays a radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored. The <VALUE> could be any integer expression.

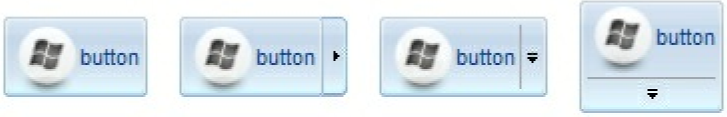


- show=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the checked item shows as selected
- spchk=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the item's sub menu is shown only if the item is checked.

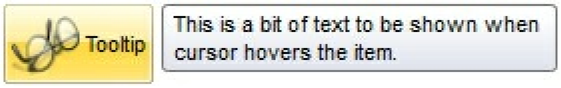


- group=<VALUE>, where <VALUE> could be a bit-or combination (+) of the following values:
    - **0** (exNoGroupPopup), No grouping is performed on the sub-menu, so the sub-items are shown to a float popup,
    - **1** (exGroupPopup), Groups and displays the sub-menu items on the current item, arranged from left to right/horizontally

  - **2** (exNoGroupPopupFrame), Prevents showing the frame around each grouping item.
  - **4** (exGroupPopupCenter), Shows the grouping popup aligned to the center of the current item.
  - **8** (exGroupPopupRight), Shows the grouping popup aligned to the right of the current item.
  - **16** (exGroupPopupEqualWidth), Shows the items that make the group of the same width
  - **32** (exGroupPopupEqualHeight), Shows the items that make the group of the same height
  - **64** (exGroupPopupFrameSolidBox), Shows a solid frame around the grouped items
  - **128** (exGroupPopupFrameThickBox), Shows a solid thick-frame around the grouped items
  - **256** (exGroupPopupVertical), Groups and displays the sub-menu items on the current item, arranged from top to bottom/vertically

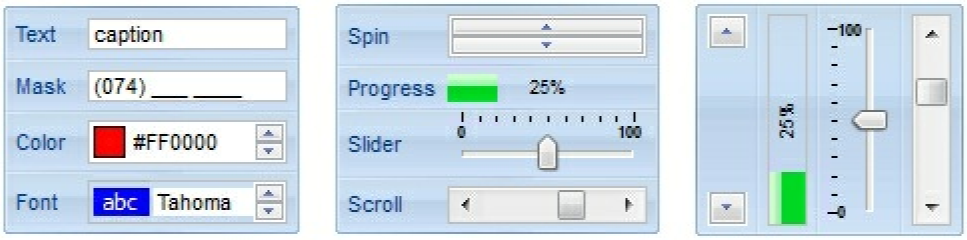


- button=<VALUE>, where <VALUE> could be a bit-or combination (+) of the following values.
  - **0** (exShowAsButtonNone), No button is shown,
  - **1** (exShowAsButton), Shows the item as a button
  - **2** (exShowAsButtonAutoSize), Fits the button to cover the item's caption instead showing on the entire item
  - **17** (exShowAsSelectButton), Shows the item as a select button, which is composed by two-fields, one indicates the default button, while the second field specifies the drop down button that displays the items in the current's sub-menu collection. The drop down button is shown to the right-side of the default button. The item must have a submenu, else no drop down is displayed.
  - **273** (exShowAsSelectButtonBottom), Shows the item as a select button, which is composed by two-fields, one indicates the default button, while the second field specifies the drop down button that displays the items in the current's sub-menu collection. The drop down button is shown to the bottom-side of the default button. The item must have a submenu, else no drop down is displayed.



This is a bit of text to be shown when cursor hovers the item.

- ttp=<VALUE>, defines the item's tooltip. The <VALUE> could be any HTML string expression. The item's tooltip is shown when the user hovers the item.



Text caption

Mask (074) ___ ____

Color #FF0000

Font abc Tahoma

- edittype=<VALUE>, associates an edit field to the item, where <VALUE> could be a combination of one or more of the following values:
    - **0** (exItemDisableEdit), No editor is assigned to the current item.
    - **1** (exItemEditText), A text-box editor is assigned to the current item.
    - **2** (exItemEditMask), A masked text-box editor is assigned to the current item.
    - **3** (exItemEditSlider), A slider editor is assigned to the current item. This can be combined with 1024.
    - **4** (exItemEditProgress), A progress editor is assigned to the current item. This can be combined with 1024.
    - 5 (exItemEditScrollBar), A scrollbar editor is assigned to the current item. This can be combined with 1024.
    - **6** (exItemEditColor), A color editor is assigned to the current item.
    - 7 (exItemEditFont), A font editor is assigned to the current item.
    - 256 (exItemEditReadOnly), specifies that the item's editor is shown as disabled. This value could be combined with one of the values from 0 to 7 or 512
    - 512 (exItemEditSpin), A spin editor is assigned to the current item. This value could be combined with one of the values from 0 to 7 or 256
    - 1024 (exItemEditVertical), The editor is shown vertically rather than horizontally. This value has effect for exItemEditSlider, exItemEditProgress or exItemEditScrollBar
- edit=<VALUE>, specifies the caption to be shown in the item's edit field, where <VALUE> could be any string
- mask=<VALUE>, specifies the mask to be applied on a masked editor. This option is valid for exItemEditMask edit. Use the float option to allow masking floating point numbers. See Masking for more information about <VALUE> of the mask option. See Masking Float for more information about <VALUE> if the float option is used.
- float=<VALUE>, Specifies whether the mask field masks a floating point number. This option is valid for exItemEditMask edit. See Masking Float for more information about <VALUE> of mask option, if the float option is used. The <VALUE> could be **0** for standard masking field or **not zero** to specify that the field is masking a floating point.
- border=<VALUE>, specifies the border to be shown on the item's edit field, where <VALUE> could be one of the following:
    - **0** (exEditBorderNone), No border is shown.
    - **-1** (exEditBorderInset), shows an inset border
    - **1** (exEditBorderSingle), shows a frame border
- editwidth=<VALUE>, specifies the width to show the edit field inside the item, where <VALUE> could be a integer expression. A negative value indicates that the field goes

to the end of the item

- min=<VALUE>, defines the minimum value of the edit field. The <VALUE> could be any integer expression, and specifies the minimum value for any slider, progress, scroll, spin, or range editor.
- max=<VALUE>, defines the maximum value of the edit field. The <VALUE> could be any integer expression, and specifies the maximum value for any slider, progress, scroll, spin, or range editor.
- tick=<VALUE>, defines where the ticks of the slider edit appear. This option is valid for exItemEditSlider edit. The <VALUE> could be one of the following values:
  - **0** (exBottomRight), The ticks are displayed on the bottom/right side.
  - **1** (exTopLeft), The ticks are displayed on the top/left side.
  - **2** (exBoth), The ticks are displayed on the both side.
  - **3** (exNoTicks), No ticks are displayed.
- freq=<VALUE>, indicates the ratio of ticks on the slider edit. This option is valid for exItemEditSlider edit. The <VALUE> could be a positive integer expression.
- ticklabel=<VALUE>, indicates the HTML label to be displayed on slider's ticks. This option is valid for exItemEditSlider edit. See Tick Label Expression for more information about <VALUE> of the ticklabel option.
- small=<VALUE>, indicates the amount by which the edit's position changes when the user presses the arrow key (left, right, or button). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive integer expression.
- large=<VALUE>, indicates the amount by which the edit's position changes when the user presses the CTRL + arrow key (CTRL + left, CTRL + right). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive integer expression.
- spin=<VALUE>, specifies the step to advance when user clicks the editor's spin.. This option is valid for exItemEditSpin edit. The <VALUE> could be a positive integer expression.
- ettp=<VALUE>, specifies the HTML tooltip to be shown when the item's value is changed. This option is valid for exItemEditSlider/exItemEditScrollBar edit. The <VALUE> could be any string expression, including built-in HTML tags

- arrow=<VALUE>. The <VALUE> could be **0** for hiding the arrow or **not zero** to show the arrow. Indicates whether an item that has a sub-menu shows or hides its popup arrow. If the <VALUE> is missing, the item shows no arrow.
- local=<VALUE>. The <VALUE> could be **0** for standard popup or **not zero** to specify that the field is a local popup. Specifies whether the item's popup is shown as local. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus.
- close=<VALUE>, Specifies the way the hosting menu is closed when the user clicks the item. If the close flag is missing, the <VALUE> is 3 (exCloseOnNonClickable), by default. The <VALUE> could be one of the following values:

    - **0** (exCloseOnClick), The popup menu is closing when the user clicks the item.
    - **1** (exCloseOnDblClick), The popup menu is closing when the user double clicks the item.
    - **2** (exCloseOnClickOutside), The popup menu is closing when the user clicks outside of the menu.
    - **3** (exCloseOnNonClickable), The popup menu is closing when the user clicks a non-clickable item (regular items). The non-clickable items is any item that's not a separator, popup, disabled or check or radio items, clicking a check-box item will makes the check box to change its state instead closing the context menu.
- popupapp=<VALUE> indicates the visual appearance of the item's submenu when the popup is shown. The <VALUE> could be a predefine value like shown bellow, or an integer expression that refers an EBN object.
    - **0** (NoBorder)
    - **1** (FlatBorder)
    - **2** (SunkenBorder)
    - **3** (RaisedBorder)
    - **4** (EtchedBorder)
    - **5** (BumpBorder)
    - **6** (ShadowBorder)
    - **7** (InsetBorder)
    - **8** (SingleBorder)
- itemsbg=<VALUE>, specifies the items background color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- itemsbghot=<VALUE>, specifies the items background color, while the cursor hovers the items, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- popupalign=<VALUE>, Indicates how the item's sub-menu is aligned relative to the parent item. The popupalign has no effect for an item that displays a select- button. The <VALUE> could be a combination of one or more of the following values:
    - **0** (exShowPopupAlignNone), The popup menu is shown on top of the item, aligned to the left (no down and right, so up and left)
    - **1** (exShowPopupAlignDown), The popup menu is shown down. If missing, the popup menu is shown up.
    - **2** (exShowPopupAlignRight), The popup menu is shown aligned to the right, else if missing, the popup menu is shown aligned to the left.
- popupat=<VALUE>, specifies the identifier of the item where the current item's submenu/popup is displayed. The <VALUE> could be any integer expression. If there is no identifier with giving value, the option has no effect.
- popupoffset=<VALUE>, specifies the offset (horizontal,vertical) to display the item's

submenu/popup relative to its default position.

- itemspad=<VALUE>, specifies the padding (space between the menu border and the item content) to display the items. The <VALUE> is a list of coordinates such as left,top,right,bottom
- visible=<VALUE>, specifies the maximum number of visible items at one time, where the <VALUE> could be any integer expression.
- tab=<VALUE>, specifies the identifier of the item/tab where the current group-popup is shown instead. The <VALUE> could be any integer expression. If there is no identifier with giving value, the option has no effect.
- itemsbgext=<VALUE>, indicates additional colors, text, images that can be displayed on the items background using the EBN String Format. The <VALUE> should be in EBN String Format. For instance, *[itemsbgext=bottom[2],bottom[16,text=`</fgcolor><fgcolor 6D6AAA>Views</fgcolor><fgcolor A0A0A0>`,align=0x21]]*, shows the Views aligned to the bottom, with a different foreground color.

Masking, (mask option)

For instance, the following input-mask (ext-phone)

!(999) 000 0000;1;;select=1,empty,overtype,warning=invalid character,invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field."

indicates the following:

- The pattern should contain 3 optional digits *999*, and 7 required digits *000 0000*, aligned to the right, *!*.
- The second part of the input mask indicates *1*, which means that all literals are included when the user leaves the field.
- The entire field is selected when it receives the focus, *select=1*
- The field supports *empty* value, so the user can leave the field with no content
- The field enters in *overtype* mode, and insert-type mode is not allowed when user pressed the Insert key
- If the user enters any invalid character, a *warning* tooltip with the message "*invalid character*" is displayed.
- If the user tries to leave the field, while the field is not validated (all 7 required digits completed), the *invalid* tooltip is shown with the message "*The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field.*" The *<%mask%>* is replaced with the first part of the input mask *!(999) 000 0000*

The four parts of an input mask, or the Mask property supports up to four parts, separated by a semicolon (;). For instance, "`Time: `00:00:00;;0;overtype,warning=<fgcolor FF0000>invalid character,beep", indicates the pattern "00:00" with the prefix Time:, the masking character being the 0, instead _, the field enters in over-type mode, insert-type mode is not allowed, and the field beeps and displays a tooltip in red with the message invalid character when the user enters an invalid character.

Input masks are made up one mandatory part and three optional parts, and each part is separated by a semicolon (;). If a part should use the semicolon (;) it must uses the \; instead

The purpose of each part is as follows:

1. The first part (pattern) is mandatory. It includes the mask characters or string (series of characters) along with placeholders and literal data such as, parentheses, periods, and hyphens.

   The following table lists the placeholder and literal characters for an input mask and explains how it controls data entry:

    - ***#***, *a digit, +, - or space (entry not required).*
    - ***0***, *a digit (0 through 9, entry required; plus [+] and minus [-] signs not allowed).*
    - ***9***, *a digit or space (entry not required; plus and minus signs not allowed).*
    - ***x***, *a lower case hexa character, [0-9],[a-f] (entry required)*
    - ***X***, *an upper case hexa character, [0-9],[A-F] (entry required)*
    - ***A***, *any letter, digit (entry required).*
    - ***a***, *any letter, digit or space (entry optional).*
    - ***L***, *any letter (entry require).*
    - ***?***, *any letter or space (entry optional).*
    - ***&***, *any character or a space (entry required).*
    - ***C***, *any character or a space (entry optional).*
    - ***>***, *any letter, converted to uppercase (entry required).*
    - ***<***, *any letter, converted to lowercase (entry required).*
    - *, *any characters combinations*
    - ***{ min,max }*** *(Range), indicates a number range. The syntax {min,max} (Range), masks a number in the giving range. The min and max values should be positive integers. For instance the mask {0,255} masks any number between 0 and 255.*
    - ***[...]*** *(Alternative), masks any characters that are contained in the [] brackets. For instance, the [abcdA-D] mask any character: a,b,c,d,A,B,C,D*
    - *\, indicates the escape character*
    - ***ť***, *(ALT + 175) causes the characters that follow to be converted to uppercase, until Ť(ALT + 174) is found.*
    - ***Ť***, *(ALT + 174) causes the characters that follow to be converted to lowercase, until ť(ALT + 175) is found.*

- ***!**, causes the input mask to fill from right to left instead of from left to right.*

  Characters enclosed in double quotation ("" or ``) marks will be displayed literally. If this part should display/use the semicolon (;) character is should be included between double quotation ("" or ``) characters or as \; (escape).

2. The second part is optional and refers to the embedded mask characters and how they are stored within the field. If the second part is set to 0 (default, exClipModeLiteralsNone), all characters are stored with the data, and if it is set to 1 (exClipModeLiteralsInclude), the literals are stored, not including the masking/placeholder characters, if 2 (exClipModeLiteralsExclude), just typed characters are stored, if 3(exClipModeLiteralsEscape), optional, required, editable and escaped entities are included. No double quoted text is included.
3. The third part of the input mask is also optional and indicates a single character or space that is used as a placeholder. By default, the field uses the underscore (_). If you want to use another character, enter it in the third part of your mask. Only the first character is considered. If this part should display/use the semicolon (;) character is should be \; (escape)
4. The forth part of the input, indicates a list of options that can be applied to input mask, separated by comma(,) character.

   The known options for the forth part are:

   - ***float**, indicates that the field is edited as a decimal number, integer. The first part of the input mask specifies the pattern to be used for grouping and decimal separators, and - if negative numbers are supported. If the first part is empty, the float is formatted as indicated by current regional settings. For instance, "##;;;float" specifies a 2 digit number in float format. The grouping, decimal, negative and digits options are valid if the float option is present.*

   - ***grouping**=value, Character used to separate groups of digits to the left of the decimal. Valid only if float is present. For instance ";;;float,grouping=" indicates that no grouping is applied to the decimal number (LOCALE_STHOUSAND)*
   - ***decimal**=value, Character used for the decimal separator. Valid only if float is present. For instance ";;;float,grouping= ,decimal=\," indicates that the decimal number uses the space for grouping digits to the left, while for decimal separator the comma character is used (LOCALE_SDECIMAL)*
   - ***negative**=value, indicates whether the decimal number supports negative numbers. The value should be 0 or 1. 1 means negative numbers are allowed.*

Else 0 or missing, the negative numbers are not accepted. Valid only if float is present.

- ***digits****=value, indicates the max number of fractional digits placed after the decimal separator. Valid only if float is present. For instance, ";;;float,digits=4" indicates a max 4 digits after decimal separator (LOCALE_IDIGITS)*
- ***password****[=value], displays a black circle for any shown character. For instance, ";;;password", specifies that the field to be displayed as a password. If the value parameter is present, the first character in the value indicates the password character to be used. By default, the * password character is used for non-TrueType fonts, else the black circle character is used. For instance, ";;;password=*", specifies that the field to be displayed as a password, and use the * for password character. If the value parameter is missing, the default password character is used.*
- ***right****, aligns the characters to the right. For instance, "(999) 999-9999;;;right" displays and masks a telephone number aligned to the right.* ***readonly****, the editor is locked, user can not update the content, the caret is available, so user can copy the text, excepts the password fields.*
- ***inserttype****, indicates that the field enters in insert-type mode, if this is the first option found. If the forth part includes also the overtype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;inserttype,overtype", indicates that the field enter in insert-type mode, and over-type mode is allowed. The "##:##;;0;inserttype", indicates that the field enter in insert-type mode, and over-type mode is not allowed.*
- ***overtype****, indicates that the field enters in over-type mode, if this is the first option found. If the forth part includes also the inserttype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;overtype,inserttype", indicates that the field enter in over-type mode, and insert-type mode is allowed. The "##:##;;0;overtype", indicates that the field enter in over-type mode, and insert-type mode is not allowed.*
- ***nocontext****, indicates that the field provides no context menu when user right clicks the field. For instance, ";;;password,nocontext" displays a password field, where the user can not invoke the default context menu, usually when a right click occurs.*
- ***beep****, indicates whether a beep is played once the user enters an invalid character. For instance, "00:00;;;beep" plays a beep once the user types in invalid character, in this case any character that's not a digit.*
- ***warning****=value, indicates the html message to be shown when the user enters an invalid character. For instance, "00:00:00;;;warning=invalid character" displays a "invalid character" tooltip once the user types in invalid character, in this case any character that's not a digit. The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword*

substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape)

- ***invalid***=*value, indicates the html message to be displayed when the user enters an inappropriate value for the field. If the value is missing or empty, the option has no effect, so no validation is performed. If the value is a not-empty value, the validation is performed. If the value is single space, no message is displayed and the field is keep opened while the value is inappropriate. For instance, "!(999) 000 0000;;;invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field." displays the "The value you entered isn't appropriate for the input mask '...' specified for this field." tooltip once the user leaves the field and it is not-valid (for instance, the field includes entities required and uncompleted). The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape). This option can be combined with empty, validateas.*
- ***validateas***=*value, specifies the additional validation is done for the current field. If value is missing or 0 (exValidateAsNone), the option has no effect. The validateas option has effect only if the invalid option specifies a not-empty value. Currently, the value can be 1 (exValidateAsDate), which indicates that the field is validated as a date. For instance, having the mask "!00/00/0000;;0;empty,validateas=1,invalid=Invalid date!,warning=Invalid character!,select=4,overtype", indicates that the field is validate as date (validateas=1).*
- ***empty***, *indicates whether the field supports empty values. This option can be used with invalid flag, which indicates that the user can leave the field if it is empty. If empty flag is present, the field displays nothing if no entity is completed (empty). Once the user starts typing characters the current mask is displayed. For instance, having the mask "!(999) 000 0000;;;empty,select=4,overtype,invalid=invalid phone number,beep", it specifies an empty or valid phone to be entered.*
- ***select***=*value, indicates what to select from the field when it got the focus. The value could be 0 (nothing, exSelectNoGotFocus), 1 (select all, exSelectAllGotFocus), 2 (select the first empty and editable entity of the field, exSelectEditableGotFocus), 3 (moves the cursor to the beginning of the first empty and editable entity of the field, exMoveEditableGotFocus), 4 (select the first empty, required and editable entity of the field, exSelectRequiredEditableGotFocus), 5 (moves the cursor to the beginning of the first empty, required and editable entity of the field, exMoveRequiredEditableGotFocus). For modes 2 and 4 the entire field is selected if no matching entity is found. For instance, "`Time:`XX:XX;;;select=1"*

indicates that the entire field (including the Time: prefix) is selected once it get the focus. The "`Time:`XX:XX;;;select=3", moves the cursor to first X, if empty, the second if empty, and so on

Experimental:
***multiline**, specifies that the field supports multiple lines.*
***rich**, specifies that the field displays a rich type editor. By default, the standard edit field is shown*
***disabled**, shows as disabled the field.*

Masking-Float, (mask, float option)

The [mask=<VALUE>] property may indicate the followings, if the [float=-1] is present

- **negative number**: if the first character in the mask is - (minus) the control supports negative numbers. Pressing the - key will toggle the sign of the number. The + sign is never displayed.
- **decimal symbol**: the last character that's different than # (digit), or 0 (zero) indicates the decimal symbol. If it is not present the control mask a floating point number without decimals.
- **thousand symbol**: the thousand symbol is the last character that's not a # (digit), 0 (zero) or it is not the decimal symbol as explained earlier, if present.
- the maximum **number of decimals** in the number (the # or 0 character after the decimal symbol)
- the maximum number of digits in the integer part (the number of # or 0 character before decimal symbol)
- the **0** character indicates **a leading-zero**. The count of 0 (zero) characters before decimal character indicates the leading-zero for integer part of the control, while the count of 0 (zero) characters after the decimal separator indicates the leading-zero for decimal part of the control. For instance, the Mask on "-###,###,##0.00", while the control's Text property is 1, the control displays 1.00, if 1.1 if displays 1.10, and if empty, the 0.00 is displayed.

If the <VALUE> property is empty, the control takes the settings for the regional options like: Decimal Symbol , No. of digits after decimal, Digit grouping symbol.

Here are few samples:

The <VALUE>"**-###.###.##0,00**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator). This format displays leading-zeros.

The <VALUE>"**-###.###.###,##**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator)

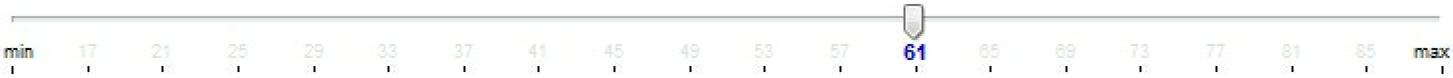
The <VALUE>"**-###,###,###.##**" filter floating point numbers a number for English settings ("." is the decimal sign, "," is the thousands separator)

The <VALUE>"**####**" indicates a max-4 digit number (positive) without a decimal symbol and without digit grouping

The <VALUE>"**-##.#**" filters a floating point number from the -99.9 to 99.9 ("." is the decimal sign, no thousands separator)

The <VALUE>"**#,###.##**" filters a floating point number from the 0 to 9,999.99 with digit grouping ("." is the decimal sign, "," is the thousands separator).

Tick Label Expression, (ticklabel option)



For instance:

- "value", shows the values for each tick.
- " (value=current ? '<font ;12><fgcolor=FF0000>' : '') + value", shows the current slider's position with a different color and font.
- "value = current ? value : """, shows the value for the current tick only.
- "(value = current ? '<b><font ;10>' : '') + (value array 'ab bc cd de ef fg gh hi ij jk kl' split ' ')" displays different captions for slider's values.

The The <VALUE> of [ticklabel] option is a formatted expression which result may include the HTML tags.

The The <VALUE> of [ticklabel] option indicates a formatting expression that may use the following predefined keywords:

- **value** gets the slider's position to be displayed
- **current** gets the current slider's value.
- **vmin** gets the slider's minimum value.
- **vmax** gets the slider's maximum value.
- **smin** gets the slider's selection minimum value.
- **smax** gets the slider's selection maximum value.

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- **/** (divide operator), priority 5
- **mod** (reminder operator), priority 5

- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

"expression ? true_part : false_part"

, while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

"expression array (c1,c2,c3,...cn)"

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')"* is equivalent with *"month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'O';10:'N';11:'D')"*.

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

"expression in (c1,c2,c3,...cn)"

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
    - #4/1/2009#, from hours 06:00 AM to 12:00 PM
    - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
    - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

    - 0 - empty (not initialized)
    - 1 - null
    - 2 - short
    - 3 - long
    - 4 - float
    - 5 - double
    - 6 - currency
    - 7 - date
    - 8 - string
    - 9 - object
    - 10 - error
    - 11 - boolean
    - 12 - variant
    - 13 - any

  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long
  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS.

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical

examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.

  - *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
  - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
    - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
    - 1 - Negative sign, number; for example, -1.1
    - 2 - Negative sign, space, number; for example, - 1.1
    - 3 - Number, negative sign; for example, 1.1-
    - 4 - Number, space, negative sign; for example, 1.1 -
  - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.
- a **split** b, splits the a using the separator b, and returns an array. For instance, the "weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' **split** ' '" gets the weekday as string. This operator can be used with the array

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

The The <VALUE> of [ticklabel] option can display labels using the following built-in HTML tags:

- **<b></b>** displays the text in **bold**.
- **<i></i>** displays the text in *italics*.
- **<u></u>** underlines the text.
- **<s></s>** ~~Strike~~-through text
- **<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>** displays text with a specified foreground color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<bgcolor=RRGGBB></bgcolor>** displays text with a specified background color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<br>** a forced line-break
- **<solidline>** The next line shows a solid-line on top/bottom side. If has no effect for a

single line caption.

- **<dotline>** The next line shows a dot-line on top/bottom side. If has no effect for a single line caption.
- **<upline>** The next line shows a solid/dot-line on top side. If has no effect for a single line caption.
- **<r>** Right aligns the text
- **<c>** Centers the text
- **<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, in UNICODE configuration. 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;

EBN String Format, (itemsbgext option)

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

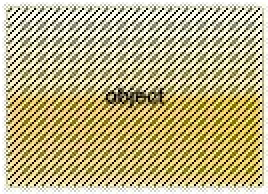
```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::=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

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

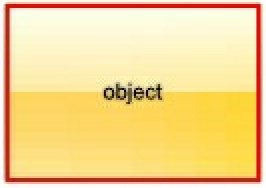
Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

Here's a few easy samples:

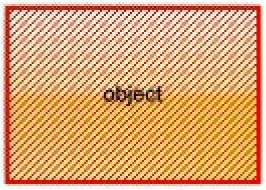
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



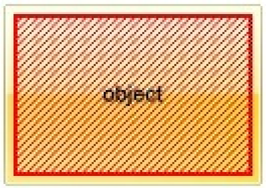
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



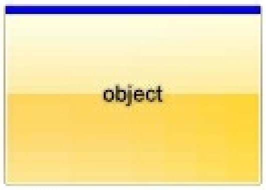
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



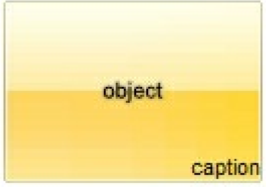
- "[[patterncolor=RGB(255,0,0)](none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



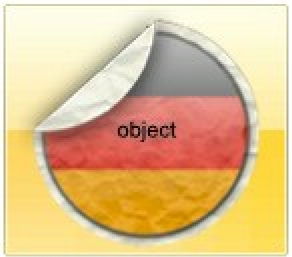
- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.



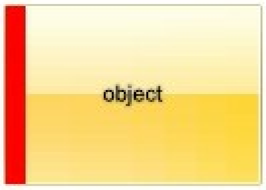
- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.



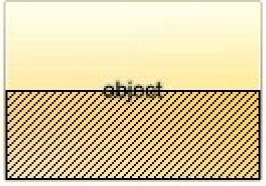
- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.



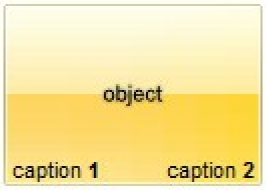
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



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



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



property Item.Underline as Boolean

Specifies whether the item's caption appears as underlined.

Type	Description
Boolean	A Boolean expression that specifies whether the item's caption is underlined.

By default, the Underline property is False. Use the Underline property to show underlined the item's caption. The Caption property indicates the HTML caption to be shown on the item. The <u> HTML tag can be used on the item's Caption property to specify different parts of the caption as underlined.

property Item.UserData as Variant

Associates an extra data to the object.

Type	Description
Variant	A VARIANT expression that indicates the item's extra data.

By default, the UserData is empty. Use the UserData property to associate any extra data to the item. Use the Caption property to specify the item's caption. Use the Tooltip property to specify the item's tooltip which can be shown when the cursor hovers the item. The Item property searches recursively the item with giving identifier/caption.

property Item.Visible as Boolean

Specifies whether the item is visible or hidden.

Type	Description
Boolean	A Boolean expression that specifies whether the item is visible or hidden.

By default, the Visible property is True. You can use the Visible property to show or hide the item. Use the Enabled property to disable an item. A disabled item shows as grayed, and it is un-selectable, so the user can select or highlight it. The Remove method removes an individual Item object giving its identifier or caption.

Items object

The Items collection supports the following properties and methods:

Name	Description
Add	Adds an Item object and returns a reference to the newly created object.
BackColor	Specifies the background color of the items.
BackgroundExt	Indicates additional colors, text, images that can be displayed on the items's background using the EBN string format.
Clear	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
HotBackColor	Specifies the hot background color of the items (when the cursor hovers the items).
item	Returns a specific Item object giving its identifier.
Padding	Specifies the padding (space between the menu border and the item content) to display the items.
PopupAppearance	Retrieves or sets the popup's appearance.
Remove	Removes a specific member from the collection.
SortOrder	Sorts the items in the submenu.
ToString	Loads or saves the Items collection using string representation.
VisibleItemsCount	Specifies the maximum number of visible items at one time.

method Items.Add (Caption as String, [ItemType as Variant], [ID as Variant])

Adds an Item object and returns a reference to the newly created object.

Type	Description
Caption as String	A String expression that specifies the HTML caption to be displayed on the item.
ItemType as Variant	An ItemTypeEnum expression that specifies the type of the item to be added.
ID as Variant	A Long expression that specifies the identifier of the item to be added.

Return	Description
Item	An Item object being created.

The Add method adds a new item to the Items collection. The ToString property loads or saves the control items from a string, so you can use the ToString method to add items too!. The Remove method removes a specified item. The Select property shows modal the context menu, and waits for the user to make the selection. The Item property gets the Item object giving its identifier or caption. The SubMenu property gets a collection of Item objects to be displayed on the sub-menu. This property returns a not-empty value, if the ItemType parameter is SubMenu. The SubControl property gets access to the Control object that holds information about the inside ActiveX or Window hosted by the item. This property returns a not-empty value, if the ItemType parameter is SubControl.

The Caption parameter supports the following HTML tags:

- **<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

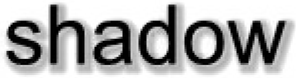
to be displayed. For instance: "Text with <font ;7><**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <font ;7><**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The <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

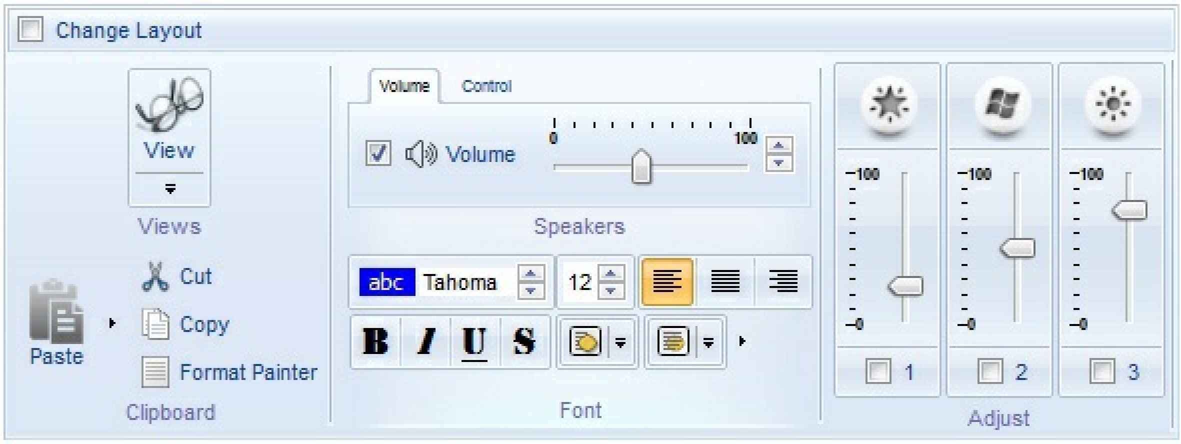
property Items.BackColor as Color

Specifies the background color of the items.

Type	Description
Color	A Color expression that specifies the items' background color. The last 7 bits in the high significant byte of the color 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 BackColor property specifies the solid color / visual appearance to be shown on the items' background (inside borders). The BackgroundExt property indicates additional colors, text, images that can be displayed on the items's background using the EBN string format. The PopupAppearance property specifies the visual appearance of the items (including the margins/borders). The HotBackColor property specifies the background color for items when cursor hovers it. The Padding property specifies the padding (space between the menu border and the item content) to display the items.

The following screen shot shows different grouping items (Clipboard, Font, Adjust) with different background appearance.



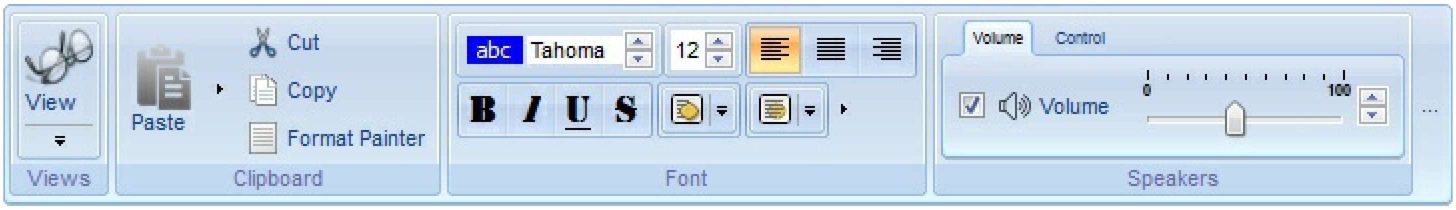
property Items.BackgroundExt as String

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

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

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

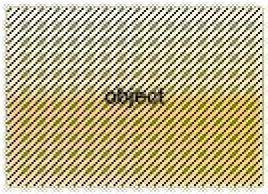
In the following screen shot the Views, Clipboard, Font and Speakers are shown using the BackgroundExt property:



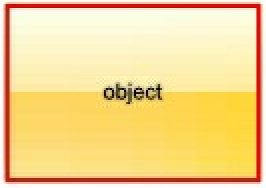
as "bottom[2],bottom[16,text=`</fgcolor><fgcolor 6D6AAA>Views</fgcolor><fgcolor A0A0A0>`,align=0x21]", shows the Views aligned to the bottom, with a different foreground color.

Easy samples:

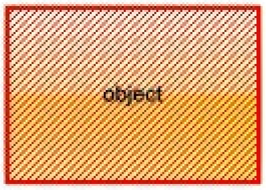
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



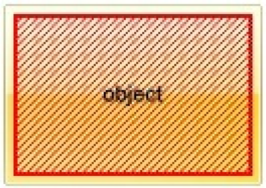
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



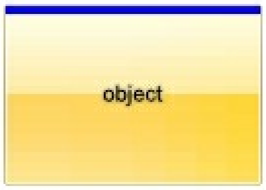
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



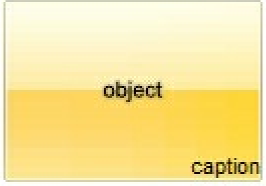
- "[[patterncolor=RGB(255,0,0)] (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



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

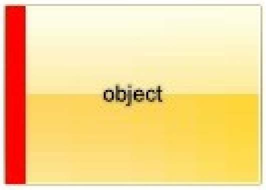


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

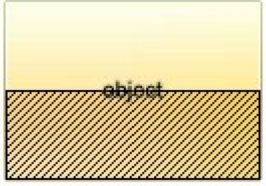


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

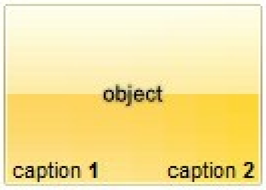
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



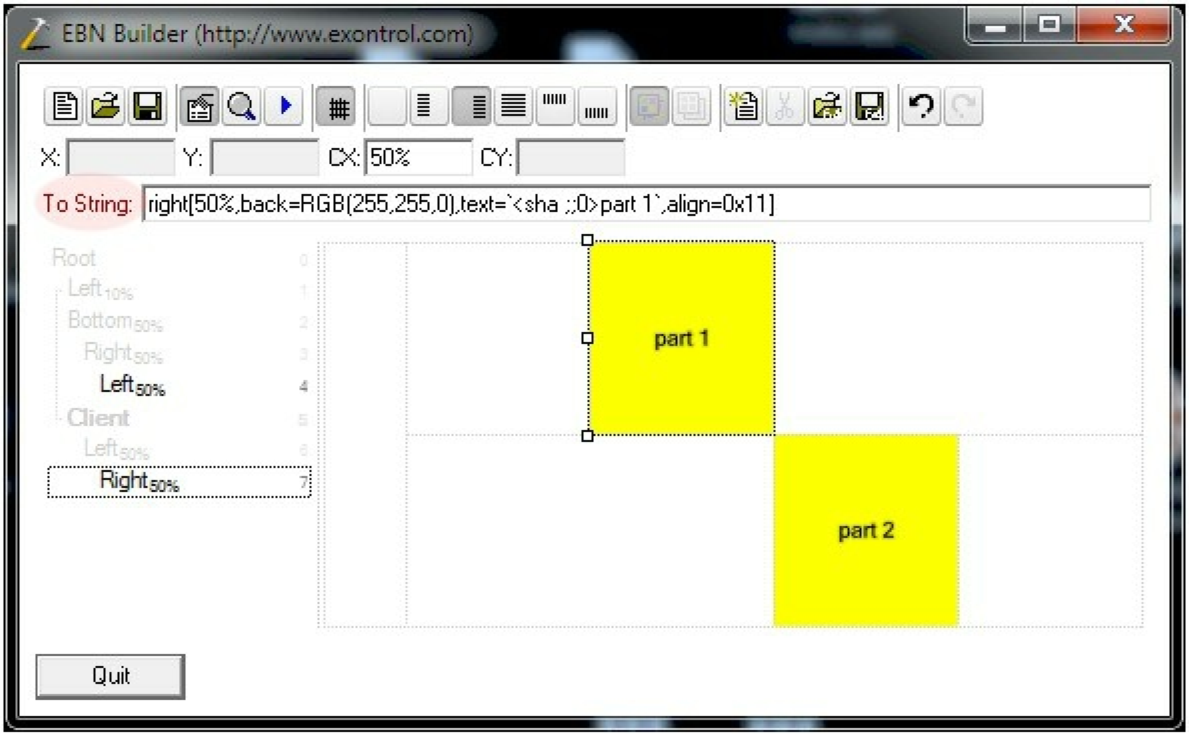
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



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



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



CX: 50%

To String: right[50%,back=RGB(255,255,0),text=`<sha ;;0>part 1`,align=0x11]

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

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

```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<client> ","] <attribute> [ "," <attributes> ] "]"
<client> ::= <expression> | <expression> "," <expression> "," <expression> "," <expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> | <wordwrap> | <align> | <pattern> | <patterncolor> | <frame> | <framethick> | <data> | <others>
<equal> ::= "="
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"  "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number> ")"
```

```
<string> ::= "`" <characters> "`" | "'" <characters> "'" | " <characters> "
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
<patterncolor> ::= "patterncolor" <equal> <color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

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

method Items.Clear ()

Removes all objects in a collection.

Type	Description

Use the Clear method to clear all elements/items in the collection. The Remove method removes an item giving its identifier.

property Items.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A Long expression that specifies the number of Item objects in the collection.

The Count property specifies the the number of Item objects in the collection. The Add method adds a new item to the Items collection, while the Remove method removes an item giving its identifier. Use the Clear method to clear all elements/items in the collection.

property Items.HotBackColor as Color

Specifies the hot background color of the items (when the cursor hovers the items).

Type	Description
Color	A Color expression that specifies the items' background color, when the cursor hovers it. The last 7 bits in the high significant byte of the color 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 HotBackColor property specifies the background color for items when cursor hovers it. The BackColor property specifies the solid color / visual appearance to be shown on the items' background (inside borders). The BackgroundExt property indicates additional colors, text, images that can be displayed on the items's background using the EBN string format. The PopupAppearance property specifies the visual appearance of the items (including the margins/borders). The Padding property specifies the padding (space between the menu border and the item content) to display the items.

property Items.item (ID as Variant) as Item

Returns a specific Item object giving its identifier.

Type	Description
ID as Variant	A Long expression that specifies the identifier of the item being requested or a String expression that specifies the caption of the item being requested.
Item	An Item object with associated identifier.

The Item property looks in the Items collection for the item with the specified identifier or caption. You can use the Item property of the context menu to recursively search for an item giving its identifier or caption. The ID property of the Item object specifies the identifier of the item. The Caption property of the Item object specifies the caption of the item. The Item property gets the first Item object being found, if multiple objects with the same identifier are found, or Nothing, if no item with associated identifier is found.

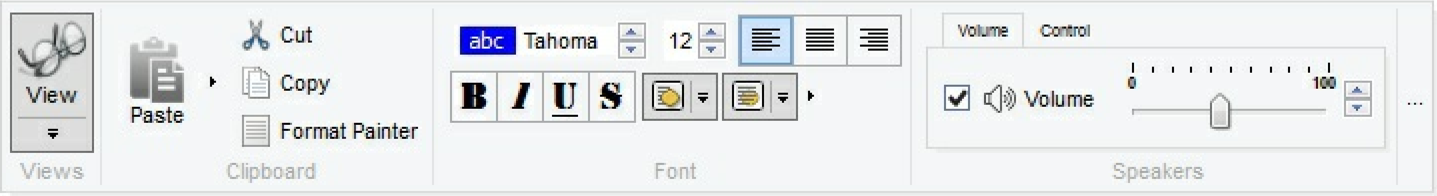
property Items.Padding as String

Specifies the padding (space between the menu border and the item content) to display the items.

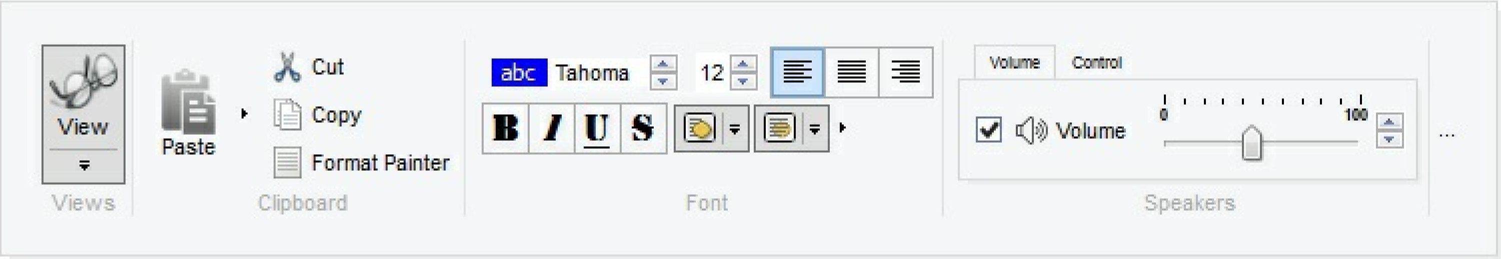
Type	Description
String	A string expression that indicates a list of 4 positive numbers separated by comma characters, which indicates the distance in pixels from margin to client, in the following format: left, top, right, bottom.

By default, the Padding property is empty (0,0,0,0). The Padding property specifies the padding (space between the menu border and the item content) to display the items. The BackgroundExt property indicates additional colors, text, images that can be displayed on the items's background using the EBN string format. When using EBN appearance, using the PopupAppearance, LocalAppearance or Appearance, the distance between margins/borders and items client area is indicated by the client object of the skin/ebn object. The Padding property specifies the padding for a particular item.

The following screen shot shows the control with no padding:



The following screen shot shows the control with padding 16, 16, 16, 16:



property Items.PopupAppearance as MenuBorderEnum

Retrieves or sets the popup's appearance.

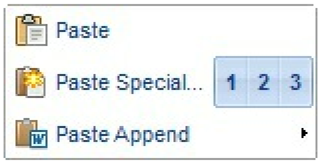
Type	Description
MenuBorderEnum	A MenuBorderEnum expression that specifies the popup's frame appearance, or a color expression whose last 7 bits in the high significant byte of the value indicates the index of the skin in the Appearance collection, being displayed as control's borders. For instance, 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 normal.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file***

By default, the PopupAppearance property is specified by the control's Appearance property. The PopupAppearance specifies a different visual appearance for the current submenu. The SubMenu property determines the items to be displayed on the popup item. The BackColor property specifies the solid color / visual appearance to be shown on the items' background (inside borders). The BackgroundExt property indicates additional colors, text, images that can be displayed on the items's background using the EBN string format. When using EBN appearance, using the PopupAppearance, LocalAppearance or Appearance, the distance between margins/borders and items client area is indicated by the client object of the skin/ebn object.

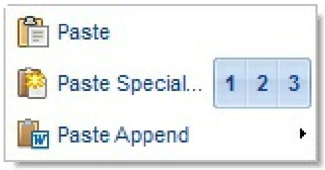
The appearance of the popup is determined by the following:

- PopupAppearance, specifies the visual appearance of the current sub-menu.
- LocalAppearance, determines the visual appearance of the popup, if it is local (ShowLocalPopup property)
- Appearance, specifies the general visual appearance of the popup items.

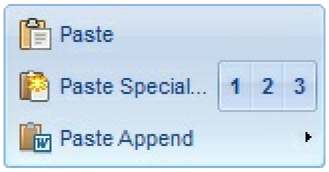
The following screen shot shows the sub-menu with different appearances:



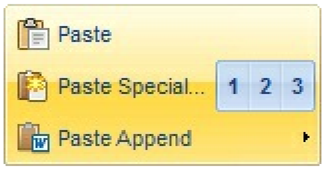
(single appearance)



(shadow appearance)



(ebn appearance)



(ebn appearance)

method Items.Remove (ID as Variant)

Removes a specific member from the collection.

Type	Description
ID as Variant	A Long expression that specifies the item to be removed. A String expression that specifies the caption of the Item to be removed

The Remove method removes an individual Item object giving its identifier or caption. The Visible property specifies whether the item is visible or hidden.

property Items.SortOrder as SubMenuSortOrderEnum

Sorts the items in the submenu.

Type	Description
SubMenuSortOrderEnum	A SubMenuSortOrderEnum expression that specifies the way the submenu displays its items.

By default, the SortOrder property is exSubMenuUnsorted, which indicates that the items are displayed on the submenu as they were added. Use the SortOrder property to sort the items to be displayed on the sub menu.

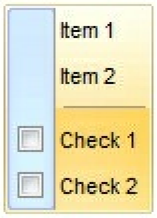
property Items.ToString as String

Loads or saves the Items collection using string representation.

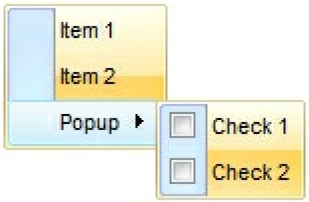
Type	Description
String	A String expression that specifies the items to be added. The list of items is separated by , (comma) character, while sub-menus are include between () parenthesis. The [] brackets indicates the options to be applied on the item

The ToString property loads or saves the control items from a string. The Add method adds a new item to the Items collection. The Remove method removes a specified item. The Select property shows modal the context menu, and waits for the user to make the selection.

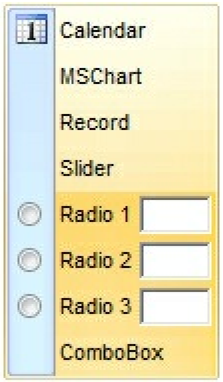
For instance, the *"Item 1,Item 2,[sep],Check 1[chk],Check 2[chk]"*, generates the following screen shot:



For instance, the *"Item 1,Item 2,Popup(Check 1[chk],Check 2[chk])"*, generates the following screen shot:



For instance, the *"Calendar[id=20][img=0],MSChart[id=30],Record[id=40],Slider[id=50],Radio 1[id=100][typ=2][edit=],Radio 2[id=101][typ=2][edit=],Radio 3[id=102][typ=2][edit=],ComboBox[id=90]"*, generates the following screen shot:



The ToString syntax in BNF notation:

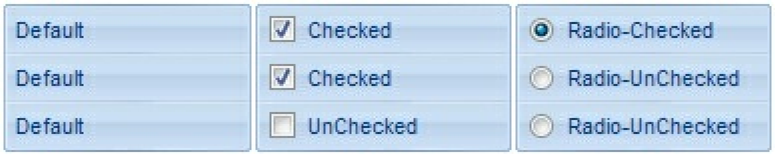
```
<ToString> ::= <ITEMS>
<ITEMS> ::= <ITEM>["("<ITEMS>")"][","<ITEMS>]
<ITEM> ::= <CAPTION>[<OPTIONS>]
<OPTIONS> ::= "["<OPTION>"]"["["<OPTIONS>"]"]
<OPTION> ::= <PROPERTY>["="<VALUE>]
<PROPERTY> ::= "img" | "himg" | "sep" | "id" | "typ" | "group" | "chk" | "button" | "align" | "spchk" | "show" | "rad" | "dis" | "showdis" | "bld" | "itl" | "stk" | "und" | "bg" | "fg" | "edittype" | "edit" | "mask" | "border" | "editwidth" | "captionwidth" | "height" | "grp" | "tfi" | "ttp" | "min" | "max" | "tick" | "freq" | "ticklabel" | "small" | "large" | "spin" | "ettp" | "float" | "close" | "local" | "popupapp" | "itemspad" | "itemsbg" | "itemsbghot" | "itemsbgext" | "visible" | "tab" | "pad" | "bghot" | "bgsel" | "bgselhot" | "arrow" | "popupalign" | "popupoffset" | "popupat"
```

where the <CAPTION> is the HTML caption to be shown on the context menu item. The <VALUE> indicates the value of giving property.

- id=<VALUE>, where <VALUE> is an integer expression, that indicates the identifier of the item.
- bg=<VALUE>, specifies the item's background color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- bghot=<VALUE>, specifies the item's background color, while the cursor hovers the item, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- bgsel=<VALUE>, specifies the item's background color, while the item is checked/selected, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- bgselhot=<VALUE>, specifies the item's background color, while the item is checked/selected and the cursor hovers it, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- fg=<VALUE>, specifies the item's foreground color, where <VALUE> could be a RGB

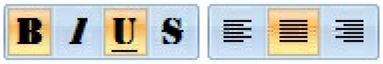
expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or a integer expression.

- sep, specifies an separator item
- dis, specifies a disabled item
- showdis=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the item shows as disabled, but it is still enabled
- bld, specifies that the item appears in bold
- itl, specifies that the item appears in italics
- stk, specifies that the item appears as strikeout
- und, specifies that the item is underlined
- align=<VALUE>, where <VALUE> could be one of the following:
  - **0** (left), to align the item's caption to the left
  - **1** (center), to center the item's caption
  - **2** (right), to align the item's caption to the right
- captionwidth=<VALUE>, specifies the width to show the HTML caption of the item. where <VALUE> could be a integer expression. A negative value indicates that no limitation is applied to the item's caption, so no truncate caption is shown
- height=<VALUE>, specifies the height to show the item, where <VALUE> could be a positive integer expression
- pad=<VALUE>, specifies the padding (space between the menu border and the item content) to display the item. The <VALUE> is a list of coordinates such as left,top,right,bottom
- img=<VALUE>, where <VALUE> is an integer expression, that indicates the index of the icon being displayed for the item.
- himg=<VALUE>, where <VALUE> indicates the key of the picture to be displayed for the item.

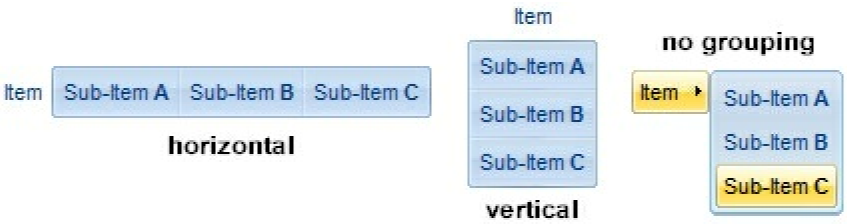


- typ=<VALUE>, where <VALUE> could be one of the following:
  - **0** for default/regular items (no check/radio button is associated with the item),
  - **1** for items that display a check/box (chk),
  - **2** to display radio buttons (rad)
- chk[=<VALUE>], where <VALUE> could be **0** for unchecked, or **not zero** for checked. The chk option makes the item to display a check box. If the <VALUE> is missing the item still displays an un-checked check box.
- rad=<VALUE>, where <VALUE> could be **0** for unchecked radio button or **not zero** to for checked radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored.
- grp=<VALUE>, defines the radio group. It should be used when you define more

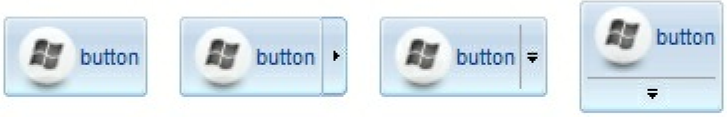
groups of radio buttons. A group of radio buttons means that only one item could be checked at one time. The rad option specifies that the item displays a radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored. The <VALUE> could be any integer expression.



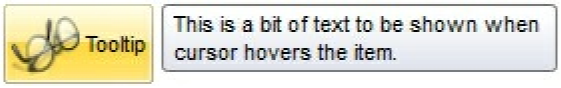
- show=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the checked item shows as selected
- spchk=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the item's sub menu is shown only if the item is checked.



- group=<VALUE>, where <VALUE> could be a bit-or combination (+) of the following values:
    - **0** (exNoGroupPopup), No grouping is performed on the sub-menu, so the sub-items are shown to a float popup,
    - **1** (exGroupPopup), Groups and displays the sub-menu items on the current item, arranged from left to right/horizontally
    - **2** (exNoGroupPopupFrame), Prevents showing the frame around each grouping item.
    - **4** (exGroupPopupCenter), Shows the grouping popup aligned to the center of the current item.
    - **8** (exGroupPopupRight), Shows the grouping popup aligned to the right of the current item.
    - **16** (exGroupPopupEqualWidth), Shows the items that make the group of the same width
    - **32** (exGroupPopupEqualHeight), Shows the items that make the group of the same height
    - **64** (exGroupPopupFrameSolidBox), Shows a solid frame around the grouped items
    - **128** (exGroupPopupFrameThickBox), Shows a solid thick-frame around the grouped items
    - **256** (exGroupPopupVertical), Groups and displays the sub-menu items on the current item, arranged from top to bottom/vertically

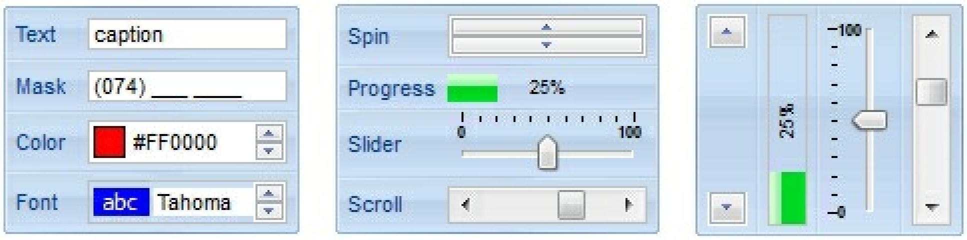


- button=<VALUE>, where <VALUE> could be a bit-or combination (+) of the following values.
  - **0** (exShowAsButtonNone), No button is shown,
  - **1** (exShowAsButton), Shows the item as a button
  - **2** (exShowAsButtonAutoSize), Fits the button to cover the item's caption instead showing on the entire item
  - **17** (exShowAsSelectButton), Shows the item as a select button, which is composed by two-fields, one indicates the default button, while the second field specifies the drop down button that displays the items in the current's sub-menu collection. The drop down button is shown to the right-side of the default button. The item must have a submenu, else no drop down is displayed.
  - **273** (exShowAsSelectButtonBottom), Shows the item as a select button, which is composed by two-fields, one indicates the default button, while the second field specifies the drop down button that displays the items in the current's sub-menu collection. The drop down button is shown to the bottom-side of the default button. The item must have a submenu, else no drop down is displayed.



This is a bit of text to be shown when cursor hovers the item.

- ttp=<VALUE>, defines the item's tooltip. The <VALUE> could be any HTML string expression. The item's tooltip is shown when the user hovers the item.



- edittype=<VALUE>, associates an edit field to the item, where <VALUE> could be a combination of one or more of the following values:
  - **0** (exItemDisableEdit), No editor is assigned to the current item.
  - **1** (exItemEditText), A text-box editor is assigned to the current item.
  - **2** (exItemEditMask), A masked text-box editor is assigned to the current item.
  - **3** (exItemEditSlider), A slider editor is assigned to the current item. This can be combined with 1024.
  - **4** (exItemEditProgress), A progress editor is assigned to the current item. This can be combined with 1024.
  - 5 (exItemEditScrollBar), A scrollbar editor is assigned to the current item. This can be combined with 1024.

- **6** (exItemEditColor), A color editor is assigned to the current item.
  - 7 (exItemEditFont), A font editor is assigned to the current item.
  - 256 (exItemEditReadOnly), specifies that the item's editor is shown as disabled. This value could be combined with one of the values from 0 to 7 or 512
  - 512 (exItemEditSpin), A spin editor is assigned to the current item. This value could be combined with one of the values from 0 to 7 or 256
  - 1024 (exItemEditVertical), The editor is shown vertically rather than horizontally. This value has effect for exItemEditSlider, exItemEditProgress or exItemEditScrollBar
- edit=<VALUE>, specifies the caption to be shown in the item's edit field, where <VALUE> could be any string
- mask=<VALUE>, specifies the mask to be applied on a masked editor. This option is valid for exItemEditMask edit. Use the float option to allow masking floating point numbers. See Masking for more information about <VALUE> of the mask option. See Masking Float for more information about <VALUE> if the float option is used.
- float=<VALUE>, Specifies whether the mask field masks a floating point number. This option is valid for exItemEditMask edit. See Masking Float for more information about <VALUE> of mask option, if the float option is used. The <VALUE> could be **0** for standard masking field or **not zero** to specify that the field is masking a floating point.
- border=<VALUE>, specifies the border to be shown on the item's edit field, where <VALUE> could be one of the following:
  - **0** (exEditBorderNone), No border is shown.
  - **-1** (exEditBorderInset), shows an inset border
  - **1** (exEditBorderSingle), shows a frame border
- editwidth=<VALUE>, specifies the width to show the edit field inside the item, where <VALUE> could be a integer expression. A negative value indicates that the field goes to the end of the item
- min=<VALUE>, defines the minimum value of the edit field. The <VALUE> could be any integer expression, and specifies the minimum value for any slider, progress, scroll, spin, or range editor.
- max=<VALUE>, defines the maximum value of the edit field. The <VALUE> could be any integer expression, and specifies the maximum value for any slider, progress, scroll, spin, or range editor.
- tick=<VALUE>, defines where the ticks of the slider edit appear. This option is valid for exItemEditSlider edit. The <VALUE> could be one of the following values:
  - **0** (exBottomRight), The ticks are displayed on the bottom/right side.
  - **1** (exTopLeft), The ticks are displayed on the top/left side.
  - **2** (exBoth), The ticks are displayed on the both side.
  - **3** (exNoTicks), No ticks are displayed.
- freq=<VALUE>, indicates the ratio of ticks on the slider edit. This option is valid for exItemEditSlider edit. The <VALUE> could be a positive integer expression.
- ticklabel=<VALUE>, indicates the HTML label to be displayed on slider's ticks. This option is valid for exItemEditSlider edit. See Tick Label Expression for more information

about <VALUE> of the ticklabel option.

- small=<VALUE>, indicates the amount by which the edit's position changes when the user presses the arrow key (left, right, or button). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive integer expression.
- large=<VALUE>, indicates the amount by which the edit's position changes when the user presses the CTRL + arrow key (CTRL + left, CTRL + right). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive integer expression.
- spin=<VALUE>, specifies the step to advance when user clicks the editor's spin.. This option is valid for exItemEditSpin edit. The <VALUE> could be a positive integer expression.
- ettp=<VALUE>, specifies the HTML tooltip to be shown when the item's value is changed. This option is valid for exItemEditSlider/exItemEditScrollBar edit. The <VALUE> could be any string expression, including built-in HTML tags

- arrow=<VALUE>. The <VALUE> could be **0** for hiding the arrow or **not zero** to show the arrow. Indicates whether an item that has a sub-menu shows or hides its popup arrow. If the <VALUE> is missing, the item shows no arrow.
- local=<VALUE>. The <VALUE> could be **0** for standard popup or **not zero** to specify that the field is a local popup. Specifies whether the item's popup is shown as local. Clicking any item inside a local popup makes the popup itself to close including all its descendent sub-menus, without closing any ascendant sub-menus.
- close=<VALUE>, Specifies the way the hosting menu is closed when the user clicks the item. If the close flag is missing, the <VALUE> is 3 (exCloseOnNonClickable), by default. The <VALUE> could be one of the following values:
    - **0** (exCloseOnClick), The popup menu is closing when the user clicks the item.
    - **1** (exCloseOnDblClick), The popup menu is closing when the user double clicks the item.
    - **2** (exCloseOnClickOutside), The popup menu is closing when the user clicks outside of the menu.
    - **3** (exCloseOnNonClickable), The popup menu is closing when the user clicks a non-clickable item (regular items). The non-clickable items is any item that's not a separator, popup, disabled or check or radio items, clicking a check-box item will makes the check box to change its state instead closing the context menu.
- popupapp=<VALUE> indicates the visual appearance of the item's submenu when the popup is shown. The <VALUE> could be a predefine value like shown bellow, or an integer expression that refers an EBN object.
    - **0** (NoBorder)
    - **1** (FlatBorder)
    - **2** (SunkenBorder)
    - **3** (RaisedBorder)
    - **4** (EtchedBorder)

    - **5** (BumpBorder)
    - **6** (ShadowBorder)
    - **7** (InsetBorder)
    - **8** (SingleBorder)
- itemsbg=<VALUE>, specifies the items background color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- itemsbghot=<VALUE>, specifies the items background color, while the cursor hovers the items, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or an integer expression to that refers an EBN object.
- popupalign=<VALUE>, Indicates how the item's sub-menu is aligned relative to the parent item. The popupalign has no effect for an item that displays a select- button. The <VALUE> could be a combination of one or more of the following values:
    - **0** (exShowPopupAlignNone), The popup menu is shown on top of the item, aligned to the left (no down and right, so up and left)
    - **1** (exShowPopupAlignDown), The popup menu is shown down. If missing, the popup menu is shown up.
    - **2** (exShowPopupAlignRight), The popup menu is shown aligned to the right, else if missing, the popup menu is shown aligned to the left.
- popupat=<VALUE>, specifies the identifier of the item where the current item's submenu/popup is displayed. The <VALUE> could be any integer expression. If there is no identifier with giving value, the option has no effect.
- popupoffset=<VALUE>, specifies the offset (horizontal,vertical) to display the item's submenu/popup relative to its default position.
- itemspad=<VALUE>, specifies the padding (space between the menu border and the item content) to display the items. The <VALUE> is a list of coordinates such as left,top,right,bottom
- visible=<VALUE>, specifies the maximum number of visible items at one time, where the <VALUE> could be any integer expression.
- tab=<VALUE>, specifies the identifier of the item/tab where the current group-popup is shown instead. The <VALUE> could be any integer expression. If there is no identifier with giving value, the option has no effect.
- itemsbgext=<VALUE>, indicates additional colors, text, images that can be displayed on the items background using the EBN String Format. The <VALUE> should be in EBN String Format. For instance, *[itemsbgext=bottom[2],bottom[16,text=`</fgcolor><fgcolor 6D6AAA>Views</fgcolor><fgcolor A0A0A0>`,align=0x21]]*, shows the Views aligned to the bottom, with a different foreground color.

Masking, (mask option)

For instance, the following input-mask (ext-phone)

!(999) 000 0000;1;;select=1,empty,overtype,warning=invalid character,invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field."

indicates the following:

- The pattern should contain 3 optional digits *999*, and 7 required digits *000 0000*, aligned to the right, *!*.
- The second part of the input mask indicates *1*, which means that all literals are included when the user leaves the field.
- The entire field is selected when it receives the focus, *select=1*
- The field supports *empty* value, so the user can leave the field with no content
- The field enters in *overtype* mode, and insert-type mode is not allowed when user pressed the Insert key
- If the user enters any invalid character, a *warning* tooltip with the message "*invalid character*" is displayed.
- If the user tries to leave the field, while the field is not validated (all 7 required digits completed), the *invalid* tooltip is shown with the message "*The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field.*" The *<%mask%>* is replaced with the first part of the input mask *!(999) 000 0000*

The four parts of an input mask, or the Mask property supports up to four parts, separated by a semicolon (;). For instance, "`Time: `00:00:00;;0;overtype,warning=<fgcolor FF0000>invalid character,beep", indicates the pattern "00:00" with the prefix Time:, the masking character being the 0, instead _, the field enters in over-type mode, insert-type mode is not allowed, and the field beeps and displays a tooltip in red with the message invalid character when the user enters an invalid character.

Input masks are made up one mandatory part and three optional parts, and each part is separated by a semicolon (;). If a part should use the semicolon (;) it must uses the \; instead

The purpose of each part is as follows:

1. The first part (pattern) is mandatory. It includes the mask characters or string (series of characters) along with placeholders and literal data such as, parentheses, periods, and hyphens.

   The following table lists the placeholder and literal characters for an input mask and explains how it controls data entry:

   - **#**, a digit, +, - or space (entry not required).
   - **0**, a digit (0 through 9, entry required; plus [+] and minus [-] signs not allowed).
   - **9**, a digit or space (entry not required; plus and minus signs not allowed).
   - **x**, a lower case hexa character, [0-9],[a-f] (entry required)
   - **X**, an upper case hexa character, [0-9],[A-F] (entry required)
   - **A**, any letter, digit (entry required).
   - **a**, any letter, digit or space (entry optional).
   - **L**, any letter (entry require).
   - **?**, any letter or space (entry optional).
   - **&**, any character or a space (entry required).
   - **C**, any character or a space (entry optional).
   - **>**, any letter, converted to uppercase (entry required).
   - **<**, any letter, converted to lowercase (entry required).
   - *****, any characters combinations
   - **{ min,max }** (Range), indicates a number range. The syntax {min,max} (Range), masks a number in the giving range. The min and max values should be positive integers. For instance the mask {0,255} masks any number between 0 and 255.
   - **[...]** (Alternative), masks any characters that are contained in the [] brackets. For instance, the [abcdA-D] mask any character: a,b,c,d,A,B,C,D
   - \, indicates the escape character
   - **ť**, (ALT + 175) causes the characters that follow to be converted to uppercase, until Ť(ALT + 174) is found.
   - **Ť**, (ALT + 174) causes the characters that follow to be converted to lowercase, until ť(ALT + 175) is found.
   - **!**, causes the input mask to fill from right to left instead of from left to right.

   Characters enclosed in double quotation ("" or ``) marks will be displayed literally. If this part should display/use the semicolon (;) character is should be included between double quotation ("" or ``) characters or as \; (escape).

2. The second part is optional and refers to the embedded mask characters and how they are stored within the field. If the second part is set to 0 (default, exClipModeLiteralsNone), all characters are stored with the data, and if it is set to 1 (exClipModeLiteralsInclude), the literals are stored, not including the masking/placeholder characters, if 2 (exClipModeLiteralsExclude), just typed characters are stored, if 3(exClipModeLiteralsEscape), optional, required, editable and escaped entities are included. No double quoted text is included.
3. The third part of the input mask is also optional and indicates a single character or space that is used as a placeholder. By default, the field uses the underscore (_). If you want to use another character, enter it in the third part of your mask. Only the first

character is considered. If this part should display/use the semicolon (;) character is should be \; (escape)

4. The forth part of the input, indicates a list of options that can be applied to input mask, separated by comma(,) character.

   The known options for the forth part are:

   - ***float**, indicates that the field is edited as a decimal number, integer. The first part of the input mask specifies the pattern to be used for grouping and decimal separators, and - if negative numbers are supported. If the first part is empty, the float is formatted as indicated by current regional settings. For instance, "##;;;float" specifies a 2 digit number in float format. The grouping, decimal, negative and digits options are valid if the float option is present.*

   - ***grouping**=value, Character used to separate groups of digits to the left of the decimal. Valid only if float is present. For instance ";;;float,grouping=" indicates that no grouping is applied to the decimal number (LOCALE_STHOUSAND)*
   - ***decimal**=value, Character used for the decimal separator. Valid only if float is present. For instance ";;;float,grouping= ,decimal=\," indicates that the decimal number uses the space for grouping digits to the left, while for decimal separator the comma character is used (LOCALE_SDECIMAL)*
   - ***negative**=value, indicates whether the decimal number supports negative numbers. The value should be 0 or 1. 1 means negative numbers are allowed. Else 0 or missing, the negative numbers are not accepted. Valid only if float is present.*
   - ***digits**=value, indicates the max number of fractional digits placed after the decimal separator. Valid only if float is present. For instance, ";;;float,digits=4" indicates a max 4 digits after decimal separator (LOCALE_IDIGITS)*
   - ***password**[=value], displays a black circle for any shown character. For instance, ";;;password", specifies that the field to be displayed as a password. If the value parameter is present, the first character in the value indicates the password character to be used. By default, the * password character is used for non-TrueType fonts, else the black circle character is used. For instance, ";;;password=*", specifies that the field to be displayed as a password, and use the * for password character. If the value parameter is missing, the default password character is used.*
   - ***right**, aligns the characters to the right. For instance, "(999) 999-9999;;;right" displays and masks a telephone number aligned to the right. **readonly**, the editor is locked, user can not update the content, the caret is available, so user can copy the text, excepts the password fields.*

- ***inserttype**, indicates that the field enters in insert-type mode, if this is the first option found. If the forth part includes also the overtype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;inserttype,overtype", indicates that the field enter in insert-type mode, and over-type mode is allowed. The "##:##;;0;inserttype", indicates that the field enter in insert-type mode, and over-type mode is not allowed.*
- ***overtype**, indicates that the field enters in over-type mode, if this is the first option found. If the forth part includes also the inserttype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;overtype,inserttype", indicates that the field enter in over-type mode, and insert-type mode is allowed. The "##:##;;0;overtype", indicates that the field enter in over-type mode, and insert-type mode is not allowed.*
- ***nocontext**, indicates that the field provides no context menu when user right clicks the field. For instance, ";;;password,nocontext" displays a password field, where the user can not invoke the default context menu, usually when a right click occurs.*
- ***beep**, indicates whether a beep is played once the user enters an invalid character. For instance, "00:00;;;beep" plays a beep once the user types in invalid character, in this case any character that's not a digit.*
- ***warning**=value, indicates the html message to be shown when the user enters an invalid character. For instance, "00:00:00;;;warning=invalid character" displays a "invalid character" tooltip once the user types in invalid character, in this case any character that's not a digit. The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape)*
- ***invalid**=value, indicates the html message to be displayed when the user enters an inappropriate value for the field. If the value is missing or empty, the option has no effect, so no validation is performed. If the value is a not-empty value, the validation is performed. If the value is single space, no message is displayed and the field is keep opened while the value is inappropriate. For instance, "!(999) 000 0000;;;invalid=The value you entered isn't appropriate for the input mask <b>'<%mask%>'</b> specified for this field." displays the "The value you entered isn't appropriate for the input mask '...' specified for this field." tooltip once the user leaves the field and it is not-valid (for instance, the field includes entities required and uncompleted). The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape). This option can be combined with empty, validateas.*
- ***validateas**=value, specifies the additional validation is done for the current field.*

If value is missing or 0 (exValidateAsNone), the option has no effect. The validateas option has effect only if the invalid option specifies a not-empty value. Currently, the value can be 1 (exValidateAsDate), which indicates that the field is validated as a date. For instance, having the mask "!00/00/0000;;0;empty,validateas=1,invalid=Invalid date!,warning=Invalid character!,select=4,overtype", indicates that the field is validate as date (validateas=1).

  - ***empty**, indicates whether the field supports empty values. This option can be used with invalid flag, which indicates that the user can leave the field if it is empty. If empty flag is present, the field displays nothing if no entity is completed (empty). Once the user starts typing characters the current mask is displayed. For instance, having the mask "!(999) 000 0000;;;empty,select=4,overtype,invalid=invalid phone number,beep", it specifies an empty or valid phone to be entered.*
  - ***select**=value, indicates what to select from the field when it got the focus. The value could be 0 (nothing, exSelectNoGotFocus), 1 (select all, exSelectAllGotFocus), 2 (select the first empty and editable entity of the field, exSelectEditableGotFocus), 3 (moves the cursor to the beginning of the first empty and editable entity of the field, exMoveEditableGotFocus), 4 (select the first empty, required and editable entity of the field, exSelectRequiredEditableGotFocus), 5 (moves the cursor to the beginning of the first empty, required and editable entity of the field, exMoveRequiredEditableGotFocus). For modes 2 and 4 the entire field is selected if no matching entity is found. For instance, "`Time:`XX:XX;;;select=1" indicates that the entire field (including the Time: prefix) is selected once it get the focus. The "`Time:`XX:XX;;;select=3", moves the cursor to first X, if empty, the second if empty, and so on*

Experimental:
***multiline**, specifies that the field supports multiple lines.*
***rich**, specifies that the field displays a rich type editor. By default, the standard edit field is shown*
***disabled**, shows as disabled the field.*

Masking-Float, (mask, float option)

The [mask=<VALUE>] property may indicate the followings, if the [float=-1] is present

- **negative number**: if the first character in the mask is - (minus) the control supports negative numbers. Pressing the - key will toggle the sign of the number. The + sign is never displayed.
- **decimal symbol**: the last character that's different than # (digit), or 0 (zero) indicates

the decimal symbol. If it is not present the control mask a floating point number without decimals.

- **thousand symbol**: the thousand symbol is the last character that's not a # (digit), 0 (zero) or it is not the decimal symbol as explained earlier, if present.
- the maximum **number of decimals** in the number (the # or 0 character after the decimal symbol)
- the maximum number of digits in the integer part (the number of # or 0 character before decimal symbol)
- the **0** character indicates **a leading-zero**. The count of 0 (zero) characters before decimal character indicates the leading-zero for integer part of the control, while the count of 0 (zero) characters after the decimal separator indicates the leading-zero for decimal part of the control. For instance, the Mask on "-###,###,##0.00", while the control's Text property is 1, the control displays 1.00, if 1.1 if displays 1.10, and if empty, the 0.00 is displayed.

If the <VALUE> property is empty, the control takes the settings for the regional options like: Decimal Symbol , No. of digits after decimal, Digit grouping symbol.

Here are few samples:

The <VALUE>"**-###.###.##0,00**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator). This format displays leading-zeros.

The <VALUE>"**-###.###.###,##**" filter floating point numbers a number for German settings ("," is the decimal sign, "." is the thousands separator)

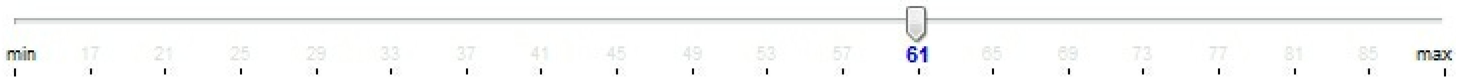
The <VALUE>"**-###,###,###.##**" filter floating point numbers a number for English settings ("." is the decimal sign, "," is the thousands separator)

The <VALUE>"**####**" indicates a max-4 digit number (positive) without a decimal symbol and without digit grouping

The <VALUE>"**-##.#**" filters a floating point number from the -99.9 to 99.9 ("." is the decimal sign, no thousands separator)

The <VALUE>"**#,###.##**" filters a floating point number from the 0 to 9,999.99 with digit grouping ("." is the decimal sign, "," is the thousands separator).

Tick Label Expression, (ticklabel option)



For instance:

- "value", shows the values for each tick.
- " (value=current ? '<font ;12><fgcolor=FF0000>' : '') + value", shows the current slider's position with a different color and font.
- "value = current ? value : ''", shows the value for the current tick only.
- "(value = current ? '<b><font ;10>' : '') + (value array 'ab bc cd de ef fg gh hi ij jk kl' split ' ')" displays different captions for slider's values.

The The <VALUE> of [ticklabel] option is a formatted expression which result may include the HTML tags.

The The <VALUE> of [ticklabel] option indicates a formatting expression that may use the following predefined keywords:

- **value** gets the slider's position to be displayed
- **current** gets the current slider's value.
- **vmin** gets the slider's minimum value.
- **vmax** gets the slider's maximum value.
- **smin** gets the slider's selection minimum value.
- **smax** gets the slider's selection maximum value.

The supported binary arithmetic operators are:

- * (multiplicity operator), priority 5
- **/** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)

- **>=** (greater or equal operator)
- **>** (greater operator)

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for is

  "expression ? true_part : false_part"

  , while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *"%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')"* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

  "expression array (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')"* is equivalent with *"month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'O';10:'N';11:'D')"*.

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

  "expression in (c1,c2,c3,...cn)"

  , where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *"value in (11,22,33,44,13)"* is equivalent with *"(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)"*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the *in* operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

"expression switch (default,c1,c2,c3,...,cn)"

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the "%0 switch ('not found',1,4,7,9,11)" gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

"expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)"

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the "*date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: "*date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)*" statement indicates the working hours for dates as follows:

-
    - #4/1/2009#, from hours 06:00 AM to 12:00 PM
    - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
    - #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions.

Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. For instance type(%0) = 8 specifies the cells that contains string values.

  Here's few predefined types:

  - 0 - empty (not initialized)
  - 1 - null
  - 2 - short
  - 3 - long
  - 4 - float
  - 5 - double
  - 6 - currency
  - 7 - date
  - 8 - string
  - 9 - object
  - 10 - error
  - 11 - boolean
  - 12 - variant
  - 13 - any
  - 14 - decimal
  - 16 - char
  - 17 - byte
  - 18 - unsigned short
  - 19 - unsigned long
  - 20 - long on 64 bits
  - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS.

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument

- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the 1000 format '' displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
  - *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
  - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
    - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
    - 1 - Negative sign, number; for example, -1.1
    - 2 - Negative sign, space, number; for example, - 1.1
    - 3 - Number, negative sign; for example, 1.1-
    - 4 - Number, space, negative sign; for example, 1.1 -
  - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)
- a **count** b (binary operator) retrieves the number of occurrences of the b in a
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result.
- a **split** b, splits the a using the separator b, and returns an array. For instance, the "weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' **split** ' '" gets the weekday as string. This operator can be used with the array

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)

- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

The The <VALUE> of [ticklabel] option can display labels using the following built-in HTML tags:

- **<b></b>** displays the text in **bold**.
- **<i></i>** displays the text in *italics*.
- **<u></u>** underlines the text.
- **<s></s>** ~~Strike~~-through text
- **<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>** displays text with a specified foreground color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<bgcolor=RRGGBB></bgcolor>** displays text with a specified background color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **<br>** a forced line-break
- **<solidline>** The next line shows a solid-line on top/bottom side. If has no effect for a single line caption.
- **<dotline>** The next line shows a dot-line on top/bottom side. If has no effect for a single line caption.
- **<upline>** The next line shows a solid/dot-line on top side. If has no effect for a single line caption.
- **<r>** Right aligns the text
- **<c>** Centers the text
- **<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, in UNICODE configuration. 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;

EBN String Format, (itemsbgext option)

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

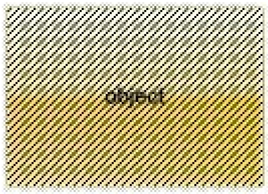
```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<client> ","] <attribute> [ "," <attributes> ] "]"
<client> ::= <expression> | <expression> "," <expression> "," <expression> ","
<expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> | <wordwrap> | <align> | <pattern> |
<patterncolor> | <frame> | <framethick> | <data> | <others>
<equal> ::= "="
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"  "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number> ")"
<string> ::= "`" <characters> "`" | "'" <characters> "'" | " <characters> "
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
```

```
<patterncolor> ::= "patterncolor" <equal> <color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

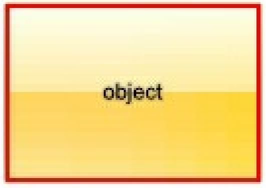
Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

Here's a few easy samples:

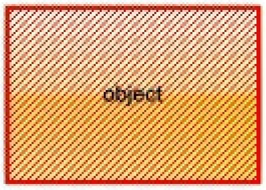
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



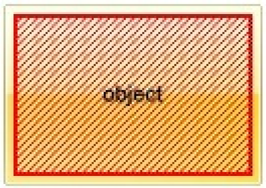
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



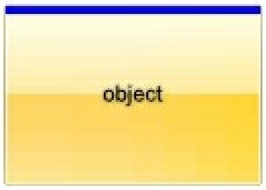
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



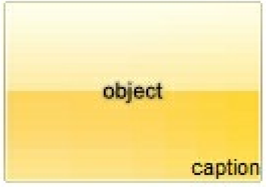
- "[[patterncolor=RGB(255,0,0)] (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



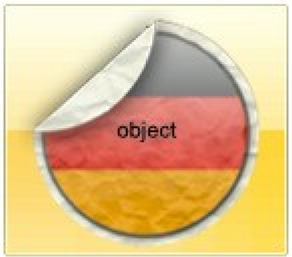
- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.



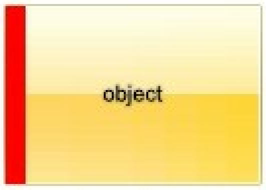
- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.



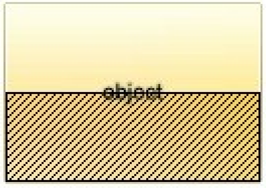
- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.



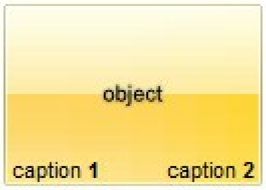
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



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



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

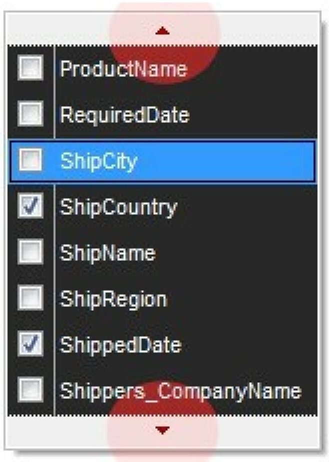

object
caption 1
caption 2

property Items.VisibleItemsCount as Long

Specifies the maximum number of visible items at one time.

Type	Description
Long	A long expression that indicates the maximum number of visible items into the popup menu.

By default, the VisibleItemCount property is 12. The control adds scroll buttons to a popup menu, if the menu contains more items than VisibleItemsCount property. Use the Visible property to specify whether an item is visible or hidden. Use the Add method to add new items to the control. The VisibleItemsCount property specifies the number of items being visible without scroll option.



OleEvent object

The OleEvent object holds information about an event fired by an ActiveX control hosted by in item that was created using the Add(,SubControl) method.

Name	Description
CountParam	Retrieves the count of the OLE event's arguments.
ID	Retrieves a long expression that specifies the identifier of the event.
Name	Retrieves the original name of the fired event.
Param	Retrieves an OleEventParam object given either the index of the parameter, or its name.
ToString	Retrieves information about the event.

property OleEvent.CountParam as Long

Retrieves the count of the OLE event's arguments.

Type	Description
Long	A long value that indicates the count of the arguments.

Use the CountParam property to count the parameters of an OLE event. Use the Name property to get the parameter name. Use the Param property to get the event's parameter. Use the Value property to specify the value of the parameter.

property OleEvent.ID as Long

Retrieves a long expression that specifies the identifier of the event.

Type	Description
Long	A Long expression that defines the identifier of the OLE event.

The identifier of the event could be used to identify a specified OLE event. Use the Name property of the OLE Event to get the name of the OLE Event. Use the ToString property to display information about an OLE event. The ToString property displays the idenfier of the event after the name of the event in two [] brackets. For instance, the ToString property gets the "KeyDown[-602](KeyCode/Short* = 9,Shift/Short = 0)" when TAB key is pressed, so the identifier of the KeyDown event being fired by the inside User editor is -602.

property OleEvent.Name as String

Retrieves the original name of the fired event.

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

Use the Name property to get the name of the event. Use the ID property to specify a specified even by its identifier. Use the ToString property to display information about fired event such us name, parameters, types and values. Use the CountParam property to count the parameters of an OLE event. Use the Param property to get the event's parameter. Use the Value property to specify the value of the parameter.

property OleEvent.Param (item as Variant) as OleEventParam

Retrieves an OleEventParam object given either the index of the parameter, or its name.

Type	Description
item as Variant	A long expression that indicates the argument's index or a a string expression that indicates the argument's name.
OleEventParam	An OleEventParam object that holds information about a parameter of an event.

Use the CountParam property to count the parameters of an OLE event. Use the Name property to get the parameter name. Use the Param property to get the event's parameter. Use the Value property to specify the value of the parameter.

property OleEvent.ToString as String

Retrieves information about the event.

Type	Description
String	A String expression that shows information about an OLE event. The ToString property gets the information as follows: Name[ID] (Param/Type = Value, Param/Type = Value, ...). For instance, "KeyDown[-602] (KeyCode/Short* = 9,Shift/Short = 0)" indicates that the KeyDown event is fired, with the identifier -602 with two parameters KeyCode as a reference to a short type with the value 8, and Shift parameter as Short type with the value 0.

Use the ToString property to display information about fired event such us name, parameters, types and values. Using the ToString property you can quickly identifies the event that you should handle in your application. Use the ID property to specify a specified even by its identifier. Use the Name property to get the name of the event. Use the Param property to access a specified parameter using its index or its name.

Displaying ToString property during the OLE Event event may show data like follows:

```
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseDown[-605](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
KeyDown[-602](KeyCode/Short* = 83,Shift/Short = 0)
KeyPress[-603](KeyAscii/Short* = 115)
Change[2]()
KeyUp[-604](KeyCode/Short* = 83,Shift/Short = 0)
MouseUp[-607](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
```

OleEventParam object

The OleEventParam holds the name and the value for an event's argument.

Name	Description
Name	Retrieves the name of the event's parameter.
Value	Retrieves the value of the event's parameter.

property OleEventParam.Name as String

Retrieves the name of the event's parameter.

Type	Description
String	A string expression that indicates the name of the event's parameter.

Use the CountParam property to count the parameters of an OLE event. Use the Name property to get the parameter name. Use the Param property to get the event's parameter. Use the Value property to specify the value of the parameter.

property OleEventParam.Value as Variant

Specifies the value of the event's parameter.

Type	Description
Variant	A variant value that indicates the value of the event's parameter.

Use the CountParam property to count the parameters of an OLE event. Use the Name property to get the parameter name. Use the Param property to get the event's parameter. Use the Value property to specify the value of the parameter.

ExContextMenu events

The Check property specifies whether the item displays a check-box inside the item. The Radio property specifies whether the item displays a radio-button inside the item. The AllowEdit property specifies whether the item displays a text-box inside. The eXContextMenu component supports the following events:

Name	Description
CheckItem	Occurs when the user checks the item.
EditChange	Occurs when the user alters the item's text box field.
Event	Notifies the application once the control fires an event.
OleEvent	Occurs when an inside ActiveX control fires an event.
SelectItem	Occurs when the user selects the item.
UncheckItem	Occurs when the user unchecks the item.

event CheckItem (Itm as Item)

Occurs when the user checks the item.

Type	Description
Itm as Item	An Item object being checked.

The CheckItem event notifies your application once the user checks the item (the item displays a check-box or a radio-button). You can use this event to update your object once the user checks an item. The user can check or uncheck the item by clicking or pressing the SPACE key while the item is selected/highlighted. The UncheckItem event notifies your application once an item is unchecked. The Check property indicates whether the Item has associated a check box. The Checked property specifies whether the item is checked or unchecked. Use the Radio property to display a radio-button on the item. In C++ or VFP, you can use the Notifier to get notified through the WM_COMMAND message.

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

C#

```
private void CheckItem(object sender,exontrol.EXCONTEXTMENULib.item Itm)
{
}
```

VB

```
Private Sub CheckItem(ByVal sender As System.Object,ByVal Itm As
exontrol.EXCONTEXTMENULib.item) Handles CheckItem
End Sub
```

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

C#

```
private void CheckItem(object sender,
AxEXCONTEXTMENULib._IExContextMenuEvents_CheckItemEvent e)
{
}
```

C++

```
void OnCheckItem(LPDISPATCH Itm)
{
}
```

C++ Builder

```
void __fastcall CheckItem(TObject *Sender,Excontextmenulib_tlb::IItem *Itm)
{
}
```

Delphi

```
procedure CheckItem(ASender: TObject; Itm : IItem);
begin
end;
```

Delphi 8 (.NET only)

```
procedure CheckItem(sender: System.Object; e:
AxEXCONTEXTMENULib._IExContextMenuEvents_CheckItemEvent);
begin
end;
```

Powe…

```
begin event CheckItem(oleobject Itm)
end event CheckItem
```

VB.NET

```
Private Sub CheckItem(ByVal sender As System.Object, ByVal e As
AxEXCONTEXTMENULib._IExContextMenuEvents_CheckItemEvent) Handles
CheckItem
End Sub
```

VB6

```
Private Sub CheckItem(ByVal Itm As EXCONTEXTMENULibCtl.IItem)
End Sub
```

VBA

```
Private Sub CheckItem(ByVal Itm As Object)
End Sub
```

VFP

```
LPARAMETERS Itm
```

Xbas…

```
PROCEDURE OnCheckItem(oExContextMenu,Itm)
RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function CheckItem(Itm)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComCheckItem Variant llItm
    Forward Send OnComCheckItem llItm
End_Procedure
```

Visual Objects

```
METHOD OCX_CheckItem(Itm) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_CheckItem(COM _Itm)
{
}
```

XBasic

```
function CheckItem as v (Itm as OLE::Exontrol.ContextMenu.1::IItem)
end function
```

dBASE

```
function nativeObject_CheckItem(Itm)
return
```

event EditChange (Itm as Item)

Occurs when the user alters the item's text box field.

Type	Description
Itm as Item	An Item object that contains a text-box inside.

The EditChange event notifies your application once the user alters the item's text-box caption. The EditCaption property specifies the caption of the text-box being altered. Use the AllowEdit property to add a text-box inside the item, so the user can type any characters inside. The EditWidth property specifies the width of the text-box inside the item. The EditBorder property specifies the border to be shown around the item's text box. You can use the Get method to collect all items of Edit type. In C++ or VFP, you can use the Notifier to get notified through the WM_COMMAND message.

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

C#

```
private void EditChange(object sender,exontrol.EXCONTEXTMENULib.item Itm)
{
}
```

VB

```
Private Sub EditChange(ByVal sender As System.Object,ByVal Itm As
exontrol.EXCONTEXTMENULib.item) Handles EditChange
End Sub
```

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

C#

```
private void EditChange(object sender,
AxEXCONTEXTMENULib._IExContextMenuEvents_EditChangeEvent e)
{
}
```

C++

```
void OnEditChange(LPDISPATCH Itm)
{
}
```

C++ Builder

```
void __fastcall EditChange(TObject *Sender,Excontextmenulib_tlb::IItem *Itm)
{
}
```

Delphi

```
procedure EditChange(ASender: TObject; Itm : IItem);
```

```
begin
end;
```

Delphi 8 (.NET only)

```
procedure EditChange(sender: System.Object; e:
AxEXCONTEXTMENULib._IExContextMenuEvents_EditChangeEvent);
begin
end;
```

Powe…

```
begin event EditChange(oleobject Itm)
end event EditChange
```

VB.NET

```
Private Sub EditChange(ByVal sender As System.Object, ByVal e As
AxEXCONTEXTMENULib._IExContextMenuEvents_EditChangeEvent) Handles
EditChange
End Sub
```

VB6

```
Private Sub EditChange(ByVal Itm As EXCONTEXTMENULibCtl.IItem)
End Sub
```

VBA

```
Private Sub EditChange(ByVal Itm As Object)
End Sub
```

VFP

```
LPARAMETERS Itm
```

Xbas…

```
PROCEDURE OnEditChange(oExContextMenu,Itm)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComEditChange Variant llItm
   Forward Send OnComEditChange llItm
End_Procedure
```

Visual Objects

```
METHOD OCX_EditChange(Itm) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_EditChange(COM _Itm)
{
}
```

XBasic

```
function EditChange as v (Itm as OLE::Exontrol.ContextMenu.1::IItem)
end function
```

dBASE

```
function nativeObject_EditChange(Itm)
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.

Let's presume that we need to handle the BarParentChange event to change the _Cancel parameter from false to true, which fires the "Error executing code: FormActiveXControl (data source), method onEvent_BarParentChange called with invalid parameters." We need to know the identifier of the BarParentChange event (each event has an unique identifier and it is static, defined in the control's type library). If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print exgantt1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
```

```
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

Now, we can see that the identifier for the BarParentChange event is 125, so we need to handle the Event event as:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
        exgantt1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

The code checks if the BarParentChange (_EventID == 125) event is fired, and changes the third parameter of the event to true. The definition for BarParentChange event can be consulted in the control's documentation or in the ActiveX explorer. So, anytime you need to access the original parameters for the event you should use the EventParam method that allows you to get or set a parameter. If the parameter is not passed by reference, you can not change the parameter's value.

Now, let's add some code to see a complex sample, so let's say that we need to prevent moving the bar from an item to any disabled item. So, we need to specify the Cancel parameter as not Items.EnableItem(NewItem), in other words cancels if the new parent is disabled. Shortly the code will be:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
        if ( !exgantt1.Items().EnableItem( exgantt1.EventParam( 2 /*NewItem*/ ) ) )
            exgantt1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

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

C#

```
private void Event(object sender,int EventID)
{
}
```

VB

```
Private Sub Event(ByVal sender As System.Object,ByVal EventID As Integer)
Handles Event
End Sub
```

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

C#

```
private void Event(object sender,
AxEXCONTEXTMENULib._IExContextMenuEvents_EventEvent e)
{
}
```

C++

```
void OnEvent(long EventID)
{
}
```

C++ Builder

```
void __fastcall Event(TObject *Sender,long EventID)
{
}
```

Delphi

```
procedure Event(ASender: TObject; EventID : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Event(sender: System.Object; e:
AxEXCONTEXTMENULib._IExContextMenuEvents_EventEvent);
begin
end;
```

Powe…

```
begin event Event(long EventID)
end event Event
```

VB.NET

```
Private Sub Event(ByVal sender As System.Object, ByVal e As
AxEXCONTEXTMENULib._IExContextMenuEvents_EventEvent) Handles Event
End Sub
```

VB6

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VBA

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VFP

```
LPARAMETERS EventID
```

Xbas…

```
PROCEDURE OnEvent(oExContextMenu,EventID)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComEvent Integer llEventID
    Forward Send OnComEvent llEventID
End_Procedure
```

Visual Objects

```
METHOD OCX_Event(EventID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Event(int _EventID)
{
}
```

XBasic

```
function Event as v (EventID as N)
end function
```

dBASE

```
function nativeObject_Event(EventID)
return
```

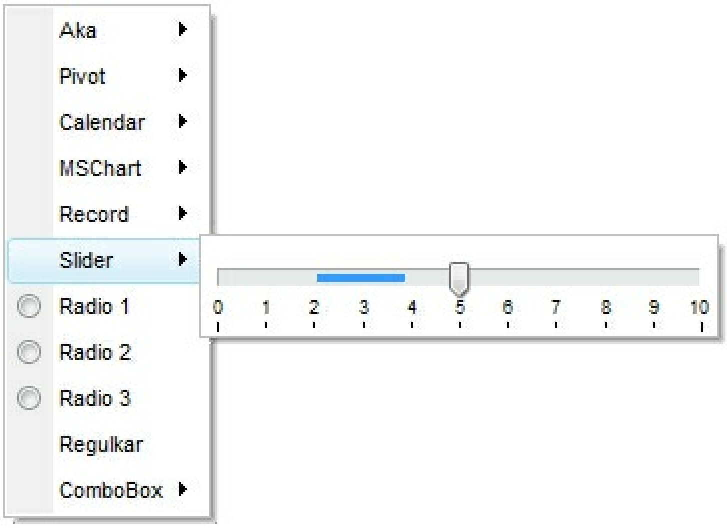
event OleEvent (Itm as Item, Ev as OleEvent)

Occurs when an inside ActiveX control fires an event.

Type	Description
Itm as Item	An Item object that contains the sub-control.
Ev as OleEvent	An OleEvent object that holds information about the fired event.

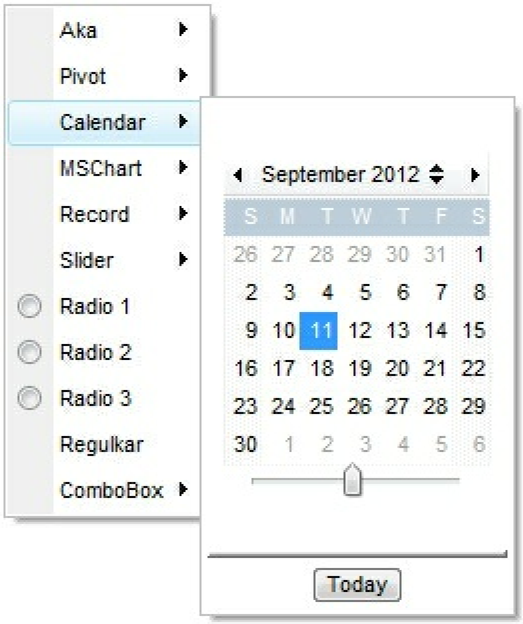
The eXContextMenu component may include sub-menus that displays any ActiveX / NET Component. The inside COM/ActiveX control fires its events through the ExContextMenu's OleEvent event. Use the ItemTypeEnum.SubControl to add an item that hosts an ActiveX inside. Use the SubControl property to access the properties to create the inside ActiveX control.

The following screen shot displays an item with an ExSlider inside:



3

The following screen shot displays an item with an ExCalendar inside:



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

C#

```
private void OleEvent(object sender,exontrol.EXCONTEXTMENULib.item
Itm,exontrol.EXCONTEXTMENULib.OleEvent Ev)
{
}
```

VB

```
Private Sub OleEvent(ByVal sender As System.Object,ByVal Itm As
exontrol.EXCONTEXTMENULib.item,ByVal Ev As
exontrol.EXCONTEXTMENULib.OleEvent) Handles OleEvent
End Sub
```

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

C#

```
private void OleEvent(object sender,
AxEXCONTEXTMENULib._IExContextMenuEvents_OleEventEvent e)
{
}
```

C++

```
void OnOleEvent(LPDISPATCH Itm,LPDISPATCH Ev)
{
}
```

C++ Builder

```
void __fastcall OleEvent(TObject *Sender,Excontextmenulib_tlb::IItem
*Itm,Excontextmenulib_tlb::IOleEvent *Ev)
{
}
```

Delphi

```
procedure OleEvent(ASender: TObject; Itm : IItem;Ev : IOleEvent);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OleEvent(sender: System.Object; e:
AxEXCONTEXTMENULib._IExContextMenuEvents_OleEventEvent);
begin
end;
```

Powe…

```
begin event OleEvent(oleobject Itm,oleobject Ev)
end event OleEvent
```

VB.NET

```
Private Sub OleEvent(ByVal sender As System.Object, ByVal e As
AxEXCONTEXTMENULib._IExContextMenuEvents_OleEventEvent) Handles
OleEvent
End Sub
```

VB6

```
Private Sub OleEvent(ByVal Itm As EXCONTEXTMENULibCtl.IItem,ByVal Ev As
EXCONTEXTMENULibCtl.IOleEvent)
End Sub
```

VBA

```
Private Sub OleEvent(ByVal Itm As Object,ByVal Ev As Object)
End Sub
```

VFP

```
LPARAMETERS Itm,Ev
```

Xbas…

```
PROCEDURE OnOleEvent(oExContextMenu,Itm,Ev)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OleEvent(Itm,Ev)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
```

```
Function OleEvent(Itm,Ev)
End Function
</SCRIPT>
```

Visual Data...

```
Procedure OnComOleEvent Variant llItm Variant llEv
    Forward Send OnComOleEvent llItm llEv
End_Procedure
```

Visual Objects

```
METHOD OCX_OleEvent(Itm,Ev) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_OleEvent(COM _Itm,COM _Ev)
{
}
```

XBasic

```
function OleEvent as v (Itm as OLE::Exontrol.ContextMenu.1::IItem,Ev as
OLE::Exontrol.ContextMenu.1::IOleEvent)
end function
```

dBASE

```
function nativeObject_OleEvent(Itm,Ev)
return
```

The following samples shows how to load an ActiveX control (Exontrol.Calendar)

VB6,VBA (MS Access, Excell...),VB.NET for /COM

```
With CreateObject("Exontrol.ContextMenu")
    With .Items.Add("Calendar",3).SubControl
        .ControlID = "Exontrol.Calendar"
        .Create
    End With
    .Select
End With
```

VB.NET

```
' Add 'exontrol.excontextmenu.dll' reference to your project.
With New exontrol.EXCONTEXTMENULib.excontextmenu()
```

```
    With .Items.Add("Calendar",3).SubControl
        .ControlID = "Exontrol.Calendar"
        .Create()
    End With
    .Select()
End With
```

C++

```
/*
    Includes the definition for CreateObject function like follows:
    #include <comdef.h>
    IUnknownPtr CreateObject( BSTR Object )
    {
        IUnknownPtr spResult;
        spResult.CreateInstance( Object );
        return spResult;
    };
*/
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXCONTEXTMENULib' for the library: 'ExContextMenu 1.0 Type Library'
    #import <ExContextMenu.dll>
    using namespace EXCONTEXTMENULib;
*/
EXCONTEXTMENULib::IExContextMenuPtr var_ExContextMenu = ::CreateObject(L"Exontrol.ContextMenu");
    EXCONTEXTMENULib::IControlPtr var_Control = var_ExContextMenu->GetItems()->Add(L"Calendar",long(3),vtMissing)->GetSubControl();
        var_Control->PutControlID(L"Exontrol.Calendar");
        var_Control->Create();
    var_ExContextMenu->Select(vtMissing,vtMissing,vtMissing);
```

C++ Builder

```
/*
    Select the Component\Import Component...\Import a Type Library,
    to import the following Type Library:
        ExContextMenu 1.0 Type Library
    TypeLib: e:\Exontrol\ExContextMenu\project\Site\ExContextMenu.dll
    to define the namespace: Excontextmenulib_tlb
*/
//#include "EXCONTEXTMENULIB_TLB.h"
Excontextmenulib_tlb::IExContextMenuPtr var_ExContextMenu =
Variant::CreateObject(L"Exontrol.ContextMenu");
    Excontextmenulib_tlb::IControlPtr var_Control = var_ExContextMenu->Items-
>Add(L"Calendar",TVariant(3),TNoParam())->SubControl;
        var_Control->ControlID = L"Exontrol.Calendar";
        var_Control->Create();
    var_ExContextMenu->Select(TNoParam(),TNoParam(),TNoParam());
```

C#

```
// Add 'exontrol.excontextmenu.dll' reference to your project.
exontrol.EXCONTEXTMENULib.excontextmenu var_ExContextMenu = new
exontrol.EXCONTEXTMENULib.excontextmenu();
    exontrol.EXCONTEXTMENULib.Control var_Control =
var_ExContextMenu.Items.Add("Calendar",3,null).SubControl;
        var_Control.ControlID = "Exontrol.Calendar";
        var_Control.Create();
    var_ExContextMenu.Select(null,null,null);
```

C# for /COM

```
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
EXCONTEXTMENULib.ExContextMenu var_ExContextMenu = new
EXCONTEXTMENULib.ExContextMenu();
    EXCONTEXTMENULib.Control var_Control =
var_ExContextMenu.Items.Add("Calendar",3,null).SubControl;
        var_Control.ControlID = "Exontrol.Calendar";
        var_Control.Create();
    var_ExContextMenu.Select(null,null,null);
```

X++ (Dynamics Ax 2009)

```
COM com_Control,com_ExContextMenu,com_Items,com_item;
anytype var_Control,var_ExContextMenu,var_Items,var_item;
;
// Add 'ExContextMenu 1.0 Type Library' reference to your project.
var_ExContextMenu = COM::createFromObject(new
EXCONTEXTMENULib.excontextmenu()); com_ExContextMenu = var_ExContextMenu;
   var_Items = COM::createFromObject(com_ExContextMenu.Items()); com_Items =
var_Items;
   var_item =
COM::createFromObject(com_Items).Add("Calendar",COMVariant::createFromInt(3));
com_item = var_item;
   var_Control = com_item.SubControl(); com_Control = var_Control;
      com_Control.ControlID("Exontrol.Calendar");
      com_Control.Create();
   com_ExContextMenu.Select();
```

Delphi 8 (.NET only)

```
with (ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMenu'))
as EXCONTEXTMENULib.ExContextMenu) do
begin
   with Items.Add('Calendar',TObject(3),Nil).SubControl do
   begin
      ControlID := 'Exontrol.Calendar';
      Create();
   end;
   Select(Nil,Nil,Nil);
end;
```

Delphi (standard)

```
with
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('Exontrol.ContextMen
 as EXCONTEXTMENULib_TLB.ExContextMenu) do
begin
   with Items.Add('Calendar',OleVariant(3),Null).SubControl do
   begin
      ControlID := 'Exontrol.Calendar';
```

```
        Create();
    end;
    Select(Null,Null,Null);
end;
```

VFP

```
with CreateObject("Exontrol.ContextMenu")
    with .Items.Add("Calendar",3).SubControl
        .ControlID = "Exontrol.Calendar"
        .Create
    endwith
    .Select()
endwith
```

XBasic (Alpha Five)

```
' Occurs when the user presses and then releases the left mouse button over
the control.
function Click as v ()
    Dim oPivot as P
    Dim var_Control as P
    Dim var_ExContextMenu as P
    oPivot = topparent:CONTROL_ACTIVEX1.activex
    var_ExContextMenu = OLE.Create("Exontrol.ContextMenu")
        var_Control = var_ExContextMenu.Items.Add("Calendar",3).SubControl
            var_Control.ControlID = "Exontrol.Calendar"
            var_Control.Create()
        var_ExContextMenu.Select()
end function

Dim oPivot as P

oPivot = topparent:CONTROL_ACTIVEX1.activex
```

Visual Objects

```
local var_ExContextMenu as IExContextMenu
// Generate Source for 'ExContextMenu 1.0 Type Library' server from Tools\Automation Server...
var_ExContextMenu := IExContextMenu{"Exontrol.ContextMenu"}
	var_Control := var_ExContextMenu:Items:Add("Calendar",3,nil):SubControl
		var_Control:ControlID := "Exontrol.Calendar"
		var_Control:Create()
	var_ExContextMenu:Select(nil,nil,nil)
```

event SelectItem (Itm as Item)

Occurs when the user selects the item.

Type	Description
Itm as Item	An Item object being clicked.

The SelectItem event notifies your application once the user clicks an item. By default, the eXContextMenu's Select method is modal, so the component waits for the user to select an item which ID is returned by the method once the drop down is closed. In other words, you do not need to handle the SelectItem event to get the item being selected/clicked, instead the Select method returns the ID of the Item being clicked. The CloseOnClick property specifies when the user closes the context menu. By default, the context menu is closed if the user clicks outside of the menu, clicks a regular item (a regular item is an item that includes no sub-menu, and no check or radio buttons). In C++ or VFP, you can use the Notifier to get notified through the WM_COMMAND message.

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

C#

```
private void SelectItem(object sender,exontrol.EXCONTEXTMENULib.item Itm)
{
}
```

VB

```
Private Sub SelectItem(ByVal sender As System.Object,ByVal Itm As
exontrol.EXCONTEXTMENULib.item) Handles SelectItem
End Sub
```

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

C#

```
private void SelectItem(object sender,
AxEXCONTEXTMENULib._IExContextMenuEvents_SelectItemEvent e)
{
}
```

C++

```
void OnSelectItem(LPDISPATCH Itm)
{
}
```

C++ Builder

```
void __fastcall SelectItem(TObject *Sender,Excontextmenulib_tlb::IItem *Itm)
{
}
```

Delphi

```
procedure SelectItem(ASender: TObject; Itm : IItem);
begin
end;
```

Delphi 8 (.NET only)

```
procedure SelectItem(sender: System.Object; e:
AxEXCONTEXTMENULib._IExContextMenuEvents_SelectItemEvent);
begin
end;
```

Powe…

```
begin event SelectItem(oleobject Itm)
end event SelectItem
```

VB.NET

```
Private Sub SelectItem(ByVal sender As System.Object, ByVal e As
AxEXCONTEXTMENULib._IExContextMenuEvents_SelectItemEvent) Handles
SelectItem
End Sub
```

VB6

```
Private Sub SelectItem(ByVal Itm As EXCONTEXTMENULibCtl.IItem)
End Sub
```

VBA

```
Private Sub SelectItem(ByVal Itm As Object)
End Sub
```

VFP

```
LPARAMETERS Itm
```

Xbas…

```
PROCEDURE OnSelectItem(oExContextMenu,Itm)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComSelectItem Variant llItm
    Forward Send OnComSelectItem llItm
End_Procedure
```

Visual Objects

```
METHOD OCX_SelectItem(Itm) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_SelectItem(COM _Itm)
{
}
```

XBasic

```
function SelectItem as v (Itm as OLE::Exontrol.ContextMenu.1::IItem)
end function
```

dBASE

```
function nativeObject_SelectItem(Itm)
return
```

event UncheckItem (Itm as Item)

Occurs when the user unchecks the item.

Type	Description
Itm as Item	An Item object being un-checked.

The UncheckItem event notifies your application once an item is unchecked (the item displays a check-box or a radio-button). The CheckItem event notifies your application once the user checks the item. You can use this event to update your object once the user checks an item. The user can check or uncheck the item by clicking or pressing the SPACE key while the item is selected/highlighted. The Check property indicates whether the Item has associated a check box. The Checked property specifies whether the item is checked or unchecked. Use the Radio property to display a radio-button on the item. In C++ or VFP, you can use the Notifier to get notified through the WM_COMMAND message.

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

C#

```
private void UncheckItem(object sender,exontrol.EXCONTEXTMENULib.item Itm)
{
}
```

VB

```
Private Sub UncheckItem(ByVal sender As System.Object,ByVal Itm As
exontrol.EXCONTEXTMENULib.item) Handles UncheckItem
End Sub
```

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

C#

```
private void UncheckItem(object sender,
AxEXCONTEXTMENULib._IExContextMenuEvents_UncheckItemEvent e)
{
}
```

C++

```
void OnUncheckItem(LPDISPATCH Itm)
{
}
```

C++ Builder

```
void __fastcall UncheckItem(TObject *Sender,Excontextmenulib_tlb::IItem *Itm)
{
}
```

Delphi

```
procedure UncheckItem(ASender: TObject; Itm : IItem);
begin
end;
```

Delphi 8 (.NET only)

```
procedure UncheckItem(sender: System.Object; e:
AxEXCONTEXTMENULib._IExContextMenuEvents_UncheckItemEvent);
begin
end;
```

Powe…

```
begin event UncheckItem(oleobject Itm)
end event UncheckItem
```

VB.NET

```
Private Sub UncheckItem(ByVal sender As System.Object, ByVal e As
AxEXCONTEXTMENULib._IExContextMenuEvents_UncheckItemEvent) Handles
UncheckItem
End Sub
```

VB6

```
Private Sub UncheckItem(ByVal Itm As EXCONTEXTMENULibCtl.IItem)
End Sub
```

VBA

```
Private Sub UncheckItem(ByVal Itm As Object)
End Sub
```

VFP

```
LPARAMETERS Itm
```

Xbas…

```
PROCEDURE OnUncheckItem(oExContextMenu,Itm)
RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function UncheckItem(Itm)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComUncheckItem Variant llItm
    Forward Send OnComUncheckItem llItm
End_Procedure
```

Visual Objects

```
METHOD OCX_UncheckItem(Itm) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_UncheckItem(COM _Itm)
{
}
```

XBasic

```
function UncheckItem as v (Itm as OLE::Exontrol.ContextMenu.1::IItem)
end function
```

dBASE

```
function nativeObject_UncheckItem(Itm)
return
```