ExRadialMenu

The eXRadialMenu (radial or pie menu) component is similar to the Microsoft's OneNote radial menu with ability to customize the appearance and functionality. The component is designed using tree structure so an item can hold none or more children, and so any item can be browsed, and show its children around it. An item can display a collection of child items, as well as a radial slider, or any other gauge / knob control. The eXRadialMenu is written from scratch, and does not depend on Windows 7, 8, 10 and so requires no dependencies to any other third party library.

Features include:

- Floating support, or ability to use as a child control into a form or floating on the screen
- Built-in radial slider, fully customizable
- Ability to display/edit data using the eXGauge component
- Pointer support, or ability to use any graphics to point to any item (rotated)
- Easy way to fill up with items, like Item A(Child 1,Child2),Item B
- Picture/Image support (PNG, BMP, JPG, GIF, TIFF, or any other known graphical file)
- Built-in HTML support
- ToolTip support
- Keyboard and Mouse Wheel support
- and more...

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 AnchorEnum

The AnchorEnum type specifies how the object is anchored. The Caption(exLayerCaptionAnchor) / ExtraCaption(...,exLayerCaptionAnchor) property specifies how the caption is anchored. The AnchorEnum type supports the following values:

Name	Value	Description
exAnchorDock	0	The object is anchored to the host's client area.
exAnchorTop	1	The object is anchored to the top side of its host.
exAnchorBottom	2	The object is anchored to the bottom side of its host.
exAnchorLeft	4	The object is anchored to the left side of its host.
exAnchorRight	8	The object is anchored to the right side of its host.

constants AppearanceEnum

The AppearanceEnum type specifies how the control's border are shown. The Appearance property specifies the control's border. The Appearance property supports the following values:

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

constants 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.

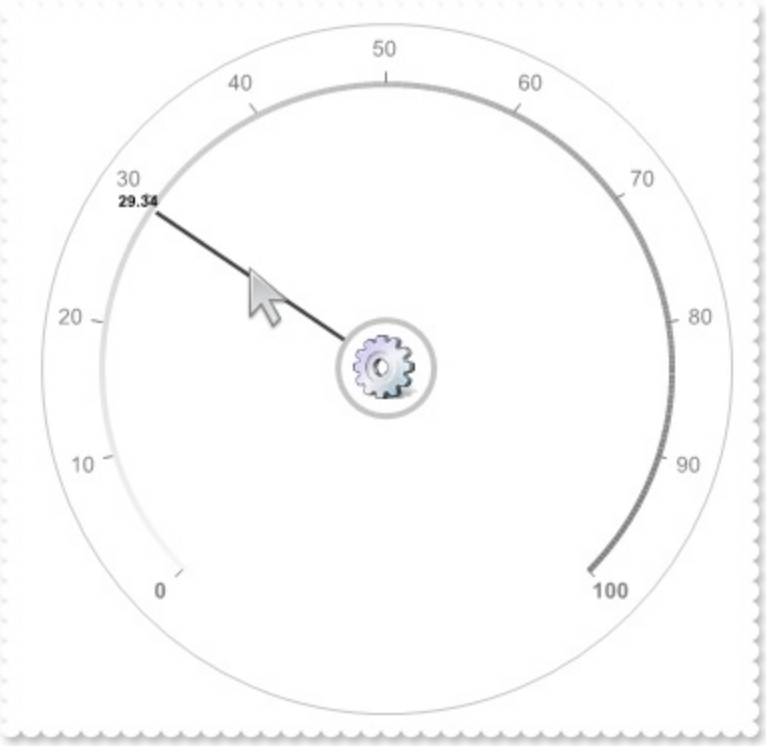
Name	Value	Description
exToolTipAppearance	64	Specifies the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The Tooltip / ToolTipTitle property indicates the item's tooltip. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ShowToolTip method to display a custom tooltip.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.

constants BrowseItemEnum

The BrowseItemEnum type specifies the type of data the control displays, when user browses for an item. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseItemEnum type supports the following values:

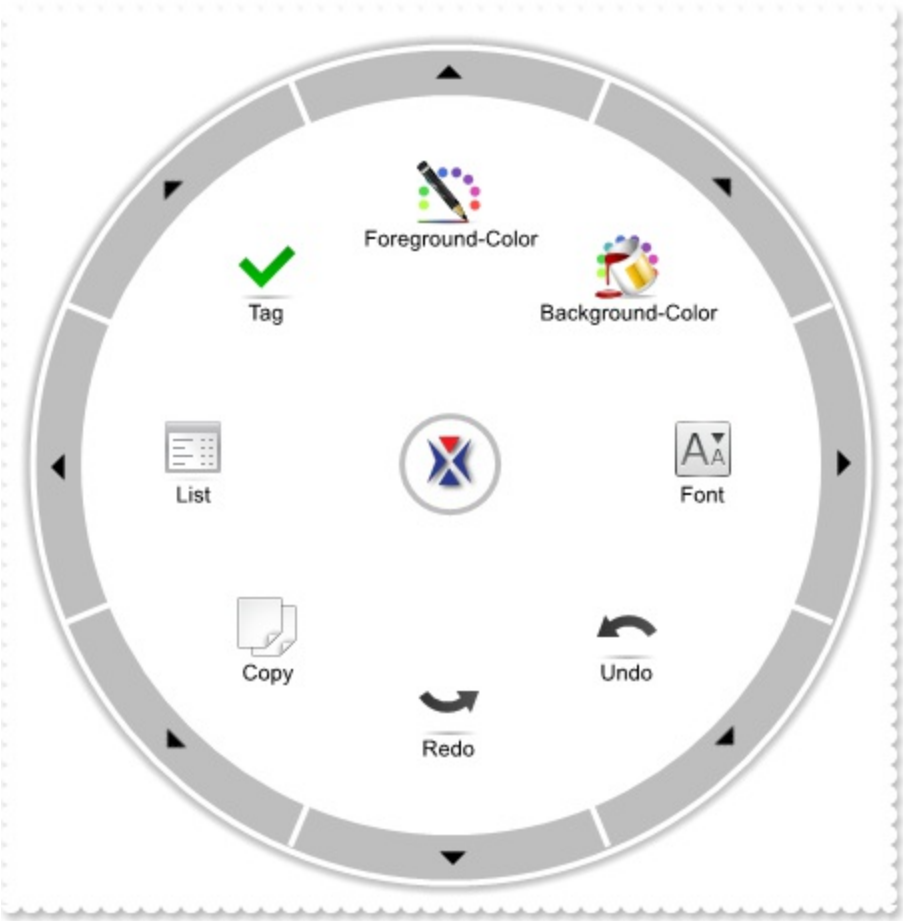
Name	Value	Description
exBrowseItemDisable	0	The item can not be browsed. Nothing happens if the user clicks the item, even it has child items or any custom control attached.
exBrowseItemChild	1	The item browses its children. The child items get displayed when the user clicks the item. 
		The item browses custom data. The custom control is being shown, when the user clicks the item.

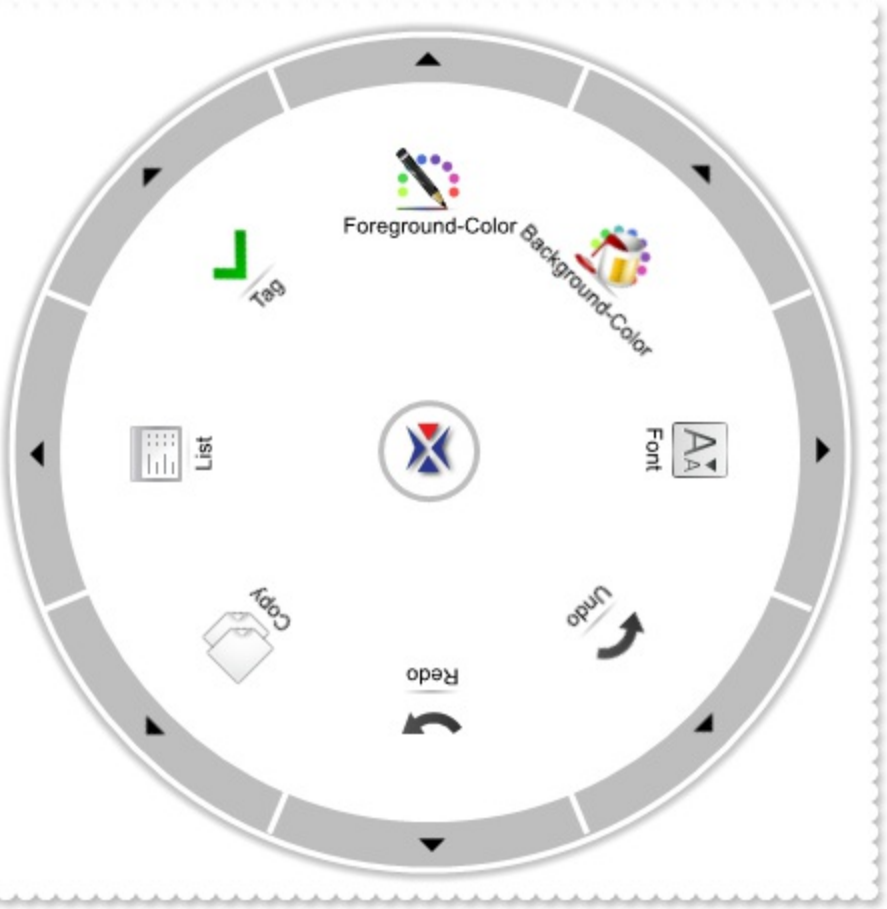
exBrowseItemCustom 2



constants DisplayRadialEnum

The DisplayRadialEnum type specifies whether the item's content is rotated. The DisplayRadial property determines how the item is displayed on the radial menu. The DisplayRadialEnum type supports the following values:

Name	Value	Description
exDisplayRadialFlat	0	The item's content is shown, without rotating. 
		The item's content is rotated from 0 to 360 degree.

exDisplayRadialRotated	1	
		The item's content is rotated from 0 to 180 degree.
exDisplayRadialRotated180	2	
		The item's content is rotated from 0 to 270 degree.

exDisplayRadialRotated270	3	
exDisplayRadialRotated90	4	The item's content is rotated.

constants LayerUpdateEnum

The LayerUpdateEnum type specifies the way the control clips its content. The control support transparent form, or in other words, displaying the control's itself without its form behind.

Currently, the control supports two type of clippings:

- by layering, using the LayerUpdate property

The LayerUpdateEnum type supports the following values:

Name	Value	Description
exLayerUpdateControl	0	By default, the control updates its content.
exLayerUpdateParent	1	Updates the parent's device, to clip the control inside.
exLayerUpdateScreen	2	Updates the screen's device, to clip the control inside.

constants PictureDisplayEnum

Specifies how a picture object is displayed. The PictureDisplayEnum type supports the following values:

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 PropertyLayerCaptionEnum

The PropertyLayerCaptionEnum type holds properties of the HTML caption that can be displayed on the control. Any of the following properties can be used to display a HTML caption:

- Caption property specifies the caption to be shown on the control's foreground.
- ExtraCaption property specifies any extra caption to be shown on the control's foreground.

The PropertyLayerCaptionEnum type supports the following value:

Name	Value	Description
exLayerCaption	0	Indicates the HTML caption to be displayed on the caption. By default, the exLayerCaption is empty. You can use the exLayerCaptionWordWrap to display the caption on multiple lines. The exLayerCaption supports built-in HTML format as listed here. *(string expression)*
exLayerCaptionBackColor	1	Indicates the caption's background color. By default, the exLayerCaptionBackColor property is -1, which indicates that no background color is applied. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. You can use the <bgcolor> HTML tag in the exLayerCaption to specify a different background color for a portion of the text. 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. *(long expression)*
exLayerCaptionForeColor	2	Indicates the caption's foreground color. By default, the exLayerCaptionForeColor property is -1, which indicates that no foreground color is applied. You can use the <fgcolor> HTML tag in the exLayerCaption to specify a different foreground

		color for a portion of the text. *(long expression)*
exLayerCaptionAnchor	3	Specifies the side of the host where the caption is anchored. By default, the exLayerCaptionAnchor property is 1 (exAnchorTop), that indicates that the caption is anchored to the top side of its host. You can use the exLayerCaptionLeft, exLayerCaptionTop, exLayerCaptionWidth and exLayerCaptionHeight to display the caption at a different position relative to its original position. *(AnchorEnum type).*
exLayerCaptionLeft	4	Specifies the expression to determine the x-position to show the caption, relative to its current position. By default, the exLayerCaptionLeft property is "0", which indicates that the caption is displayed at it's original position (horizontal axis), determined by the exLayerCaptionAnchor. You can use the exLayerCaptionAnchor property to anchor the caption to a different side of the host. The property supports the following keywords: • **twidth**, indicates the width required to fully display the caption • **theight**, indicates the height required to fully display the caption • **width**, indicates the width of the control (if it is applied to the control's foreground Caption or ExtraCaption) • **height**, indicates the height control (if it is applied to the control's foreground Caption or ExtraCaption) The property supports predefined constants, operators and functions as described here . *(string expression)*
		Specifies the expression to determine the y-position

exLayerCaptionTop	5	to show the caption, relative to its current position. By default, the exLayerCaptionTop property is "0", which indicates that the caption is displayed at it's original position (vertical axis), determined by the exLayerCaptionAnchor. You can use the exLayerCaptionAnchor property to anchor the caption to a different side of the host. The property supports the following keywords: • **twidth**, indicates the width required to fully display the caption • **theight**, indicates the height required to fully display the caption • **width**, indicates the width of the control (if it is applied to the control's foreground Caption or ExtraCaption) • **height**, indicates the height control (if it is applied to the control's foreground Caption or ExtraCaption) The property supports predefined constants, operators and functions as described here . *(string expression)*
exLayerCaptionWidth	6	Specifies the expression to determine the width to show the caption, relative to its current width. By default, the exLayerCaptionWidth property is "twidth", which indicates that the caption is displayed on its full width. You can use the exLayerCaptionAnchor property to anchor the caption to a different side of the host. The property supports the following keywords: • **twidth**, indicates the width required to fully display the caption • **theight**, indicates the height required to fully display the caption • **width**, indicates the width of the control (if it is applied to the control's foreground Caption or ExtraCaption)

		• **height**, indicates the height control (if it is applied to the control's foreground Caption or ExtraCaption) The property supports predefined constants, operators and functions as described here . *(string expression)*
exLayerCaptionHeight	7	Specifies the expression to determine the height to show the caption, relative to its current height. By default, the exLayerCaptionHeight property is "theight", which indicates that the caption is displayed on its full height. You can use the exLayerCaptionAnchor property to anchor the caption to a different side of the host. The property supports the following keywords: • **twidth**, indicates the width required to fully display the caption • **theight**, indicates the height required to fully display the caption • **width**, indicates the width of the control (if it is applied to the control's foreground Caption or ExtraCaption) • **height**, indicates the height control (if it is applied to the control's foreground Caption or ExtraCaption) The property supports predefined constants, operators and functions as described here . *(string expression)*
exLayerCaptionWordWrap	8	Indicates whether a multiline caption automatically wraps words to the beginning of the next line when necessary. By default, the exLayerCaptionWordWrap property is False. *(boolean expression)*
		Indicates Unlimited options to show any HTML text,

exLayerCaptionBackgroundEx	9	images, colors, EBNs, patterns, frames anywhere on the layer's background, using EBN String Format. A short description of the EBN String Format is described here, or a full description of the EBN String Format can be found here. *(string expression)*
exLayerCaptionVisibleFront	10	Specifies whether the caption is shown in front. By default, the exLayerCaptionVisibleFront property is True, which indicates that the caption is shown in front. Use the exLayerCaptionVisibleFront property to display the caption on the layer's background, if the exLayerCaptionVisibleFront property is False. *(boolean expression)*

The exLayerCaption supports built-in HTML tags as follow:

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

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

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

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

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

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

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

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so

on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

variable := expression

where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance,

the *upper("mihai")* returns "MIHAI"

- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"

- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

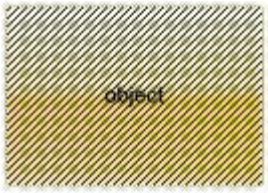
The EBN String Format syntax in BNF notation is defined like follows:

```
<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"
```

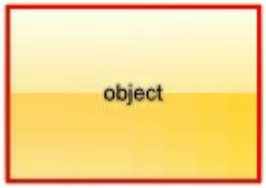
Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

Easy samples:

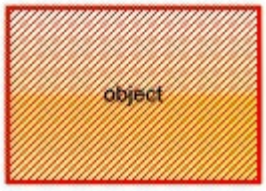
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



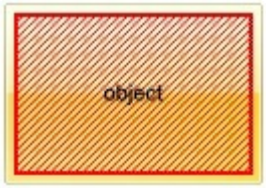
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



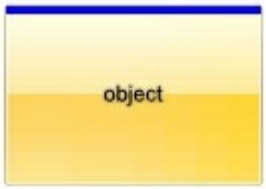
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



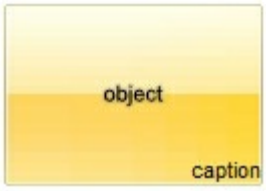
- "[[patterncolor=RGB(255,0,0)] (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



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

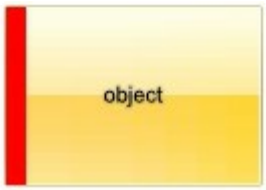


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

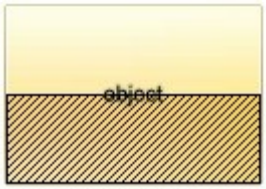


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

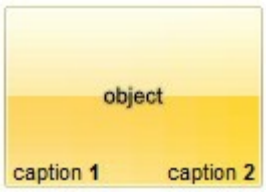
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



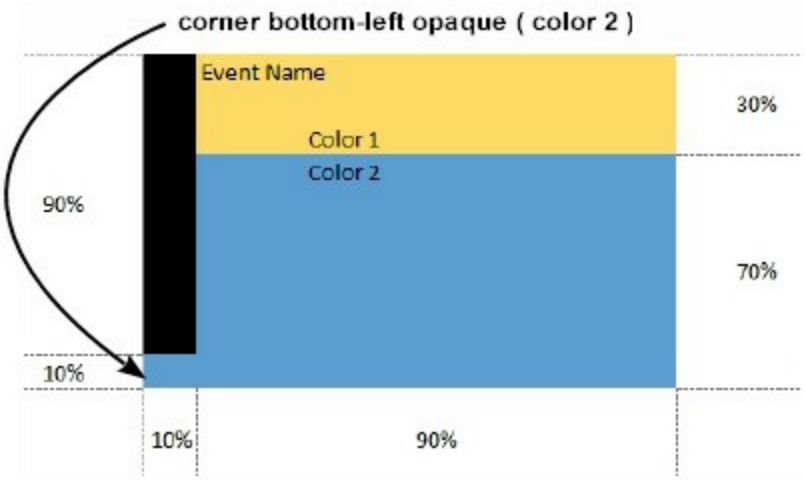
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



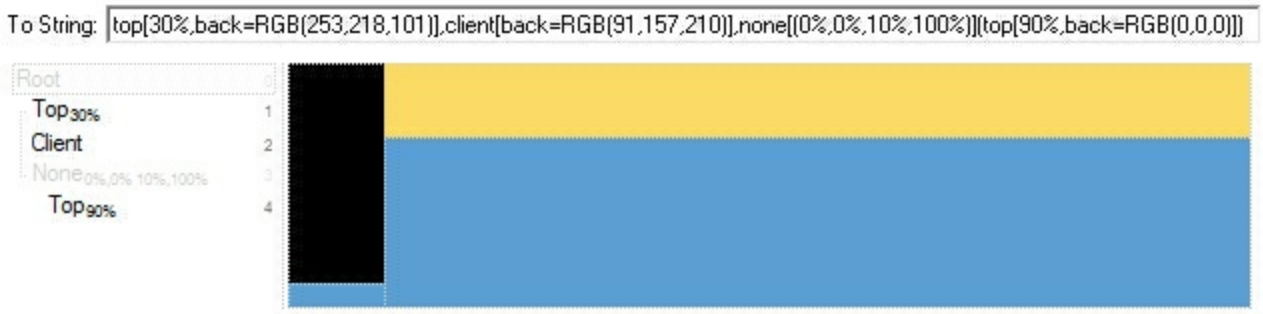
- "root[text=`caption <b>2`,align=0x22](client[text=`caption <b>1`,align=0x20])", shows the caption **1** aligned to the bottom-left side, and the caption **2** to the bottom-right side



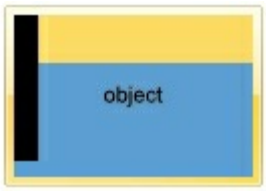
Now, lets say we have the following request to layout the colors on the objects:



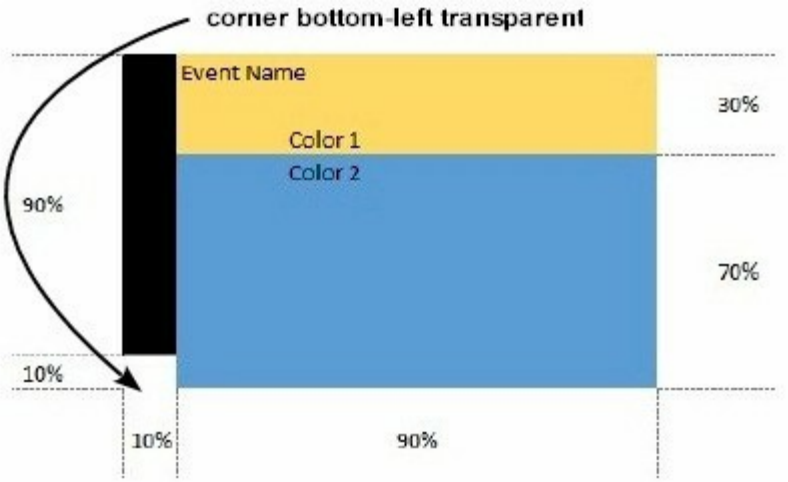
We define the BackgroundExt property such as "top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100%)](top[90%,back=RGB(0,0,0)])", and it looks as:



so, if we apply to our object we got:

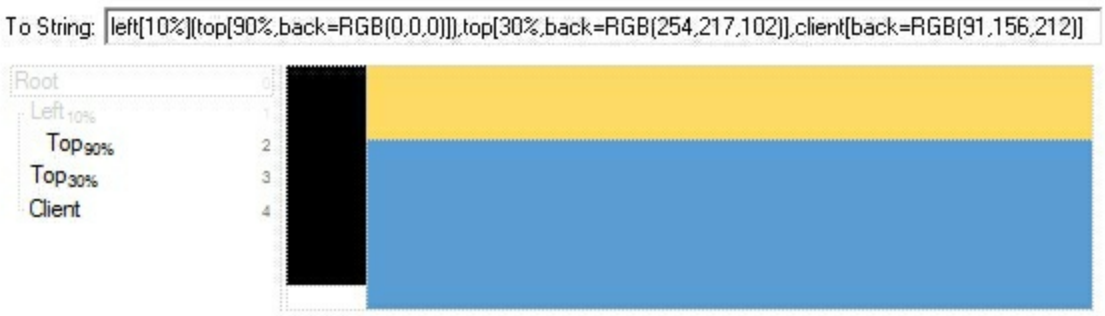


Now, lets say we have the following request to layout the colors on the objects:

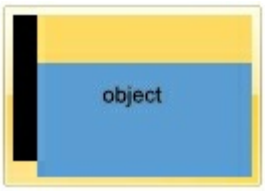


We define BackgroundExt property such as "left[10%]

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



so, if we apply to our object we got:



constants RadialCustomPropertyEnum

The RadialCustomPropertyEnum type defines the properties of the custom control. The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

The properties prefixed with:

- *exRadialCustomSlider..., should be used while the BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomSlider*
- *exRadialCustomGauge..., should be used while the BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomGauge*

The RadialCustomPropertyEnum type supports the following values:

Name	Value	Description
exRadialCustomSliderMinValue	1	By default, the exRadialCustomSliderMinValue is 0. Specifies the minimum value of the slider. (double expression)
exRadialCustomSliderMaxValue	2	By default, the exRadialCustomSliderMaxValue is 100. Specifies the maximum value. (double expression)
exRadialCustomSliderTickFrequency	3	By default, the exRadialCustomSliderTickFrequency is 10. If 0, no ticks are shown. Gets or sets the interval between tick marks. (double expression)
exRadialCustomSliderStartAngle	4	By default, the exRadialCustomSliderStartAngle is 135 degree. Indicates the angle to start the scale, from first radial line, in degree . (double expression)
exRadialCustomSliderSweepAngle	5	By default, the exRadialCustomSliderSweepAngle is 270 degree. Indicates the angle to end the scale, to second radial line, in degree. (double expression)

exRadialCustomSliderScaleColor	6	By default, the exRadialCustomSliderScaleColor is RGB(128,128,128). Gets or sets the color to display the scale. (long expression)
exRadialCustomSliderScaleAlpha	7	By default, the exRadialCustomSliderScaleAlpha is 255. Gets or sets the transparency byte to display the scale. (byte expression)
exRadialCustomSliderScaleGradient	8	By default, the exRadialCustomSliderScaleGradient is True. Specifies whether the scale is shown in gradient. (boolean expression)
exRadialCustomSliderValue	9	By default, the exRadialCustomSliderValue is 0. Gets or sets the radial slider's value. (double expression)
exRadialCustomSliderNeedleColor	10	By default, the exRadialCustomSliderNeedleColor is RGB(64,64,64). Gets or sets the color to display the needle. (long expression)
exRadialCustomSliderNeedleAlpha	11	By default, the exRadialCustomSliderNeedleAlpha is 255. Gets or sets the transparency byte to display the needle. (byte expression)
exRadialCustomSliderLocked	12	By default, the exRadialCustomSliderLocked is False. Specifies whether the radial slider is read-only or locked. (boolean expression)
		By default, the

exRadialCustomSliderShowHandCursor	13	exRadialCustomSliderShowHandCursor is True. Indicates whether the hand cursor is shown, when the cursor hovers the needle. (boolean expression)
exRadialCustomSliderAdjustValue	14	By default, the exRadialCustomSliderAdjustValue is "value". The exRadialCustomSliderAdjustValue expression supports the following keywords: • **value**, indicates the current value of the slider, exRadialCustomSliderValue • **vmin**, specifies the minimum value of the slider, exRadialCustomSliderMinValue • **vmax**, specifies the maximum value of the slider, exRadialCustomSliderMaxValue • **tick**, indicates the interval between tick marks, exRadialCustomSliderTickFrequency The expression supports constants, operators and functions defined here. Gets or sets the expression that determines the valid values of the radial slider. (string expression)
exRadialCustomSliderLabelTick	15	By default, the exRadialCustomSliderLabelTick is "`<fgcolor 808080>` + (((value = vmin) or (value = vmax)) ? `<b>` : ``) + (value format `0`) ". The exRadialCustomSliderLabelTick expression supports the following keywords: • **value**, indicates the current value of the slider, exRadialCustomSliderValue • **vmin**, specifies the minimum value of the slider, exRadialCustomSliderMinValue • **vmax**, specifies the maximum value of the slider, exRadialCustomSliderMaxValue • **tick**, indicates the interval between tick marks,

	exRadialCustomSliderTickFrequency The expression supports constants, operators and functions defined here. Specifies the labels to be shown on tick marks. (string expression)
exRadialCustomSliderLabelValue 16	By default, the exRadialCustomSliderLabelValue is "`<b><font ;6>` + (value format ``)". The exRadialCustomSliderLabelValue expression supports the following keywords: • **value**, indicates the current value of the slider, exRadialCustomSliderValue • **vmin**, specifies the minimum value of the slider, exRadialCustomSliderMinValue • **vmax**, specifies the maximum value of the slider, exRadialCustomSliderMaxValue • **tick**, indicates the interval between tick marks, exRadialCustomSliderTickFrequency The expression supports constants, operators and functions defined here. Specifies the label to be shown on the current value. (string expression)
exRadialCustomSliderWheelAdvance 17	By default, the exRadialCustomSliderLabelValue is 1. Gets or sets a value that indicates the advancement to be added to the current value, when mouse's wheel is rotated. (double expression)
	By default, the exRadialCustomGaugeHandle is 0. Specifies the handle of the inside control to be shown instead. This property must be specified if the BrowseType property is exBrowseItemCustom,

exRadialCustomGaugeHandle 256	and BrowseCustomType property is exRadialCustomGauge. The hWnd property of the eXGauge must be passed to the BrowseCustom(exRadialCustomGaugeHandle) property. The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. (long expression)

The expression supports the following constants, operators and functions:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:**

are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1

- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the

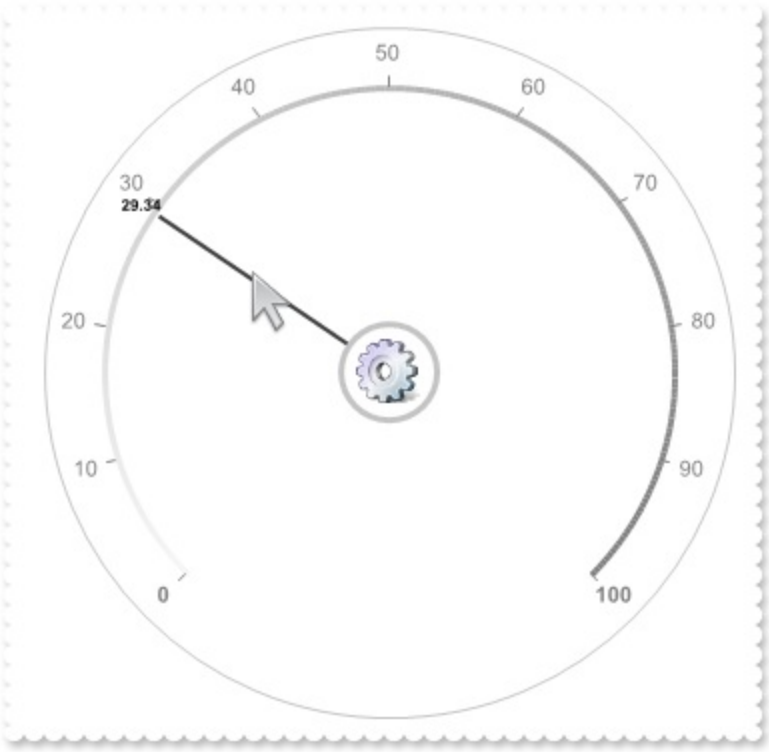
month(#12/31/1971 13:14:15#) returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

constants RadialCustomTypeEnum

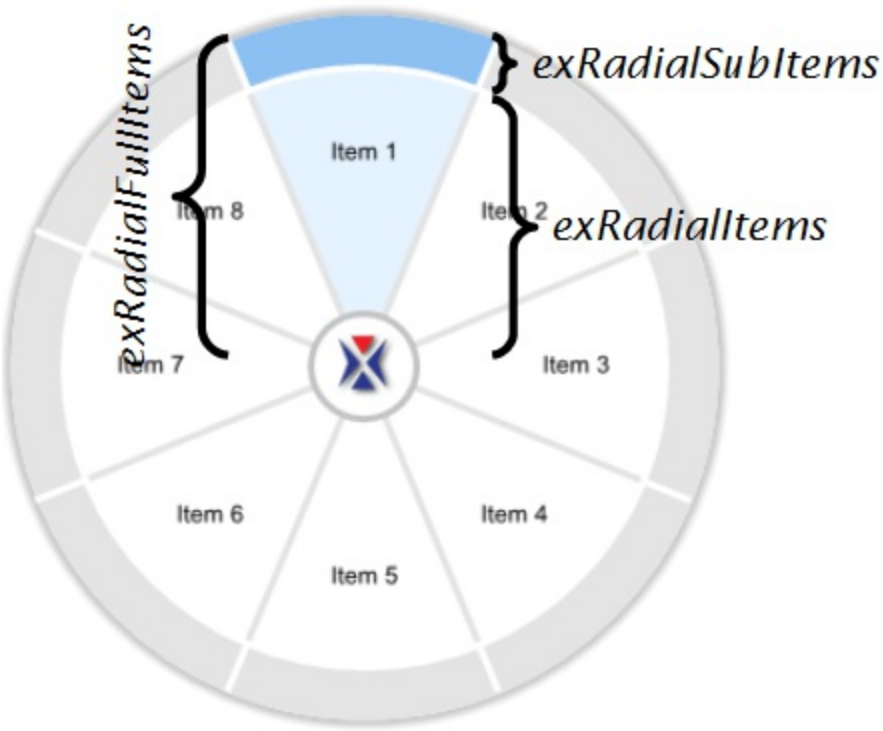
The RadialCustomTypeEnum type specifies custom controls that the radial menu can handle. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item. The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property must be specified when the BrowseType property is exBrowseItemCustom. The RadialCustomTypeEnum type supports the following values:

Name	Value	Description
exRadialCustomDisable	0	No custom control is displayed when browsing the item. By default, the child items get displayed instead. 
		A radial-slider is displayed when user browses the item. (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomSlider)

exRadialCustomSlider	16	
exRadialCustomGauge	32	A gauge control is displayed when user browses the item. (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomGauge). The control displays/edit data using the using the Exontrol's ExGauge component. 

constants RadialItemsEnum

The RadialItemsEnum type specifies the portions of the item as described in the next picture:



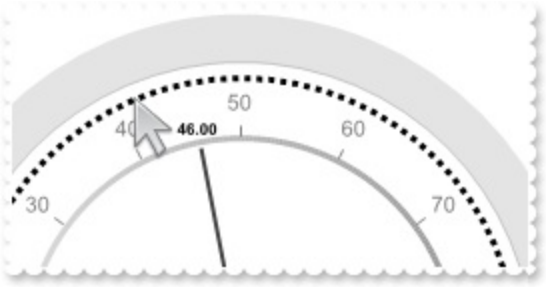
The RadialItemsEnum type supports the following values:

Name	Value	Description
exRadialItems	1	Specifies the items portion of the radial menu.
exRadialSubItems	2	Specifies the sub-items portion of the radial menu.
exRadialFullItems	3	Specifies the items and sub-items portions of the radial menu.

constants RadialLineEnum

The RadialLineEnum type specifies the radial-lines that you can change. The RadialLineColor property specifies the color to show the given radial line within the control. The RadialLineAlpha property specifies the value of alpha / opacity channel to show the giving line within the radial menu. The RadialLineSize property specifies the size to show the giving line within the radial menu. The RadialLineStyle property specifies the style to show the given radial line within the control.

The following screen shot show the exRadialCustomBorder, with a different style and size:



The RadialLineEnum type supports the following values:

Name	Value	Description
exRadialBorder	1	exRadialBorder.
exRadialSubItemsBorder	2	exRadialSubItemsBorder.
exRadialSubItemsGridLines	3	exRadialSubItemsGridLines.
exRadialCustomBorder	4	exRadialCustomBorder.
exRadialItemsGridLines	5	exRadialItemsGridLines.
exRadialParentBorder	6	exRadialParentBorder.
exRadialItemsBorder	7	exRadialItemsBorder.
exRadialHotParent	8	exRadialHotParent.
exRadialHotItem	9	exRadialHotItem.
exRadialHotSubItem	10	exRadialHotSubItem.
exRadialHotFullItem	11	exRadialHotFullItem.
exRadialSubItemsChildren	12	exRadialSubItemsChildren.
exRadialItemsChildren	13	exRadialItemsChildren.

constants RadialLineStyleEnum

The RadialLineStyleEnum type specifies the style of the radial line. The RadialLineColor property specifies the color to show the given radial line within the control. The RadialLineAlpha property specifies the value of alpha / opacity channel to show the giving line within the radial menu. The RadialLineSize property specifies the size to show the giving line within the radial menu. The RadialLineStyle property specifies the style to show the given radial line within the control. The RadialLineStyleEnum type supports the following value:

Name	Value	Description
exRadialLineSolid	0	exRadialLineSolid.
exRadialLineDash	1	exRadialLineDash.
exRadialLineDot	2	exRadialLineDot.
exRadialLineDashDot	3	exRadialLineDashDot.
exRadialLineDashDotDot	4	exRadialLineDashDotDot.

constants RadialMenuFloatEnum

The RadialMenuFloatEnum type specifies where the control's content is rendered. The Float property specifies whether the control is shown as float.

The following screen shot shows the control on a transparent form (**Float**):

The RadialMenuFloatEnum type supports the following values:

Name	Value	Description
exRadialMenuChild	0	The control is displayed as a child of a form.
exRadialMenuFloat	1	The control is displayed as float.
exRadialMenuFloatTopmost	2	The control is displayed as topmost float panel.

constants RadialMenuStateEnum

The RadialMenuStateEnum type specifies the radial menu's states. The State property specifies the state of the radial menu. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The RadialMenuStateEnum type supports the following values.

Name	Value	Description
exRadialMenuCollapsed	0	The radial menu is shown collapsed (Expanded property is False)
exRadialMenuExpandedNoItems	1	The radial menu is shown expanded, when it contains no items (Expanded property is True, Count property is 0)
exRadialMenuExpandedRootItem	2	The radial menu browses the root item, expanded, and it contains child-items (Expanded property is True, Count property is not 0)
exRadialMenuExpandedChildItem	3	The radial menu browses a child item, expanded.
exRadialMenuMissingInheritItemImage	4	Only for internal use.
exRadialMenuStateAll	-1	Indicates all states of the radial menu.

Appearance object

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

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

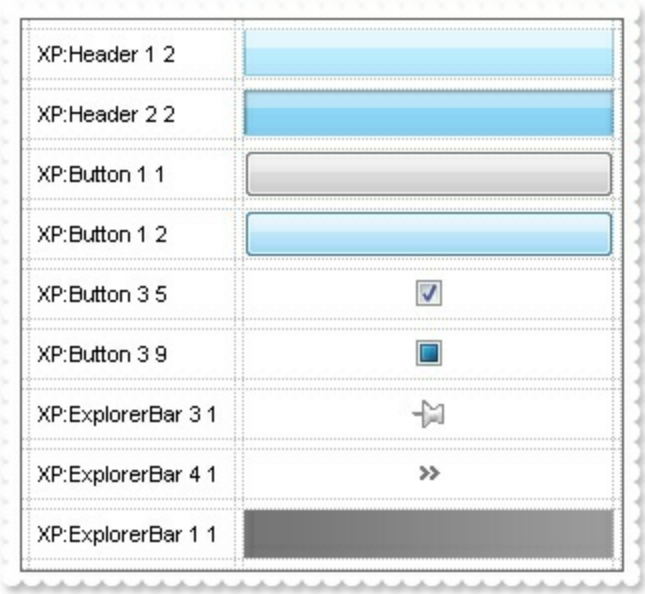
method Appearance.Add (ID as Long, Skin as Variant)

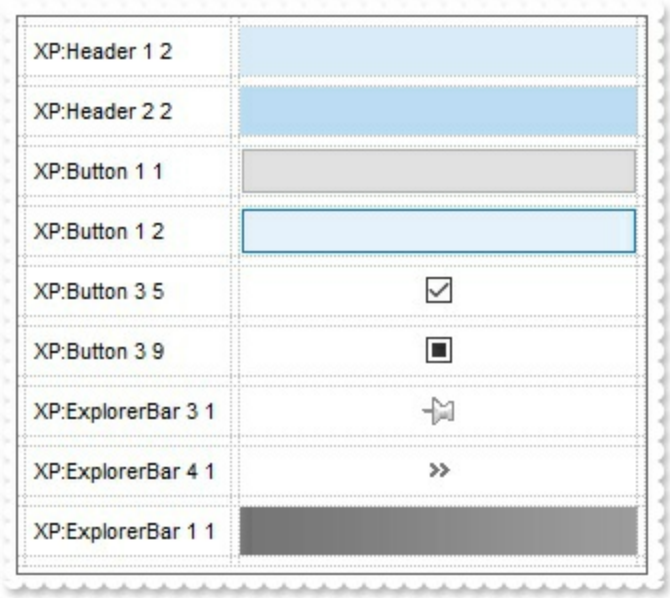
Adds or replaces a skin object to the control.

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

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

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



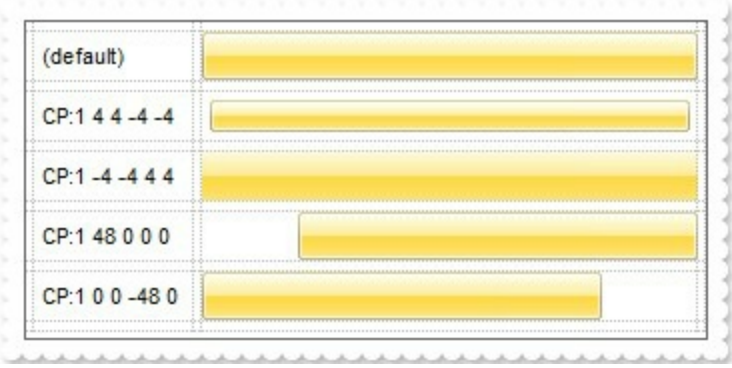


Skin as Variant

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

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

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



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

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

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

Control/ClassName	Part	States
BUTTON	BP_CHECKBOX = 3	CBS_UNCHECKED 1 CBS_UNCHECKE CBS_UNCHECKED = 3 CBS_UNCHECKED = 4 CBS_CHECKE 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_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_CHECKE 5 RBS_CHECKEDH RBS_CHECKEDPR RBS_CHECKEDDIS

	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1
COMBOBOX	CP_DROPDOWNBUTTON = 1	CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED =
EDIT	EP_CARET = 2	
	EP_EDITTEXT = 1	ETS_NORMAL = 1 2 ETS_SELECTED ETS_DISABLED = ETS_FOCUSED = 5 ETS_READONLY = ETS_ASSIST = 7
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
	EBP_HEADERCLOSE = 2	EBHC_NORMAL = EBHC_HOT = 2 EBHC_PRESSED =
	EBP_HEADERPIN = 3	EBHP_NORMAL = EBHP_HOT = 2 EBHP_PRESSED = EBHP_SELECTEDN 4 EBHP_SELECTE EBHP_SELECTEDF 6
	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSE
	EBP_NORMALGROUPBACKGROUND = 5	
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_NORMAL EBNGC_HOT = 2 EBNGC_PRESSED
	EBP_NORMALGROUPEXPAND = 7	EBNGE_NORMAL EBNGE_HOT = 2 EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
	EBP_SPECIALGROUPCOLLAPSE = 10	EBSGC_NORMAL EBSGC_HOT = 2 EBSGC_PRESSED
	EBP_SPECIALGROUPEXPAND = 11	EBSGE_NORMAL EBSGE_HOT = 2

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

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

	TP_DROPDOWNBUTTON = 2	TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTON = 3	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTONDROPDOWN = 4	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATOR = 5	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATORVERT = 6	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
TOOLTIP	TTP_BALLOON = 3	TTBS_NORMAL = 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 = 4 TUS_DISABLED = TUBS_NORMAL = TUBS_HOT = 2

	TKP_THUMBBOTTOM = 4	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 = 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 =
	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

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

	MAXBS_DISABLED
WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_VERTSCROLL = 27	VSS_NORMAL = 1 = 2 VSS_PUSHED VSS_DISABLED =
WP_VERTTHUMB = 28	VTS_NORMAL = 1 2 VTS_PUSHED = 3 VTS_DISABLED = 4

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

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

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

property Appearance.RenderType as Long

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

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

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

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

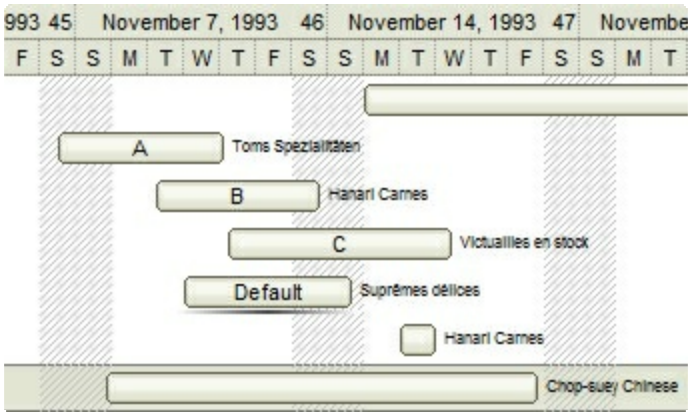
Click here to watch a movie on how you can change the colors to be applied on EBN objects.

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

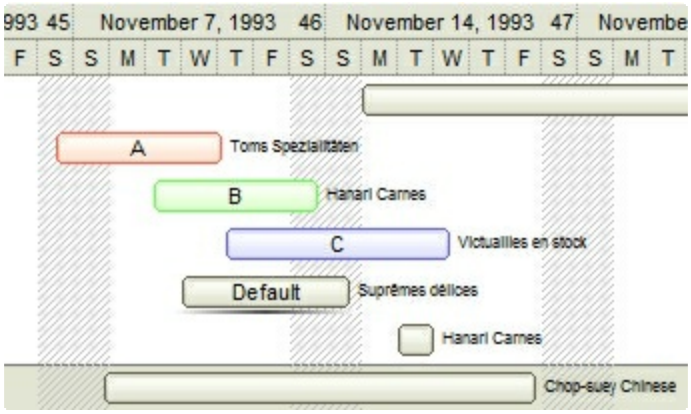
- "A" in Red (RGB(255,0,0), for instance the bar's property exBarColor is 0x10000FF
- "B" in Green (RGB(0,255,0), for instance the bar's property exBarColor is 0x100FF00
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance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

The RenderType property could be one of the following:

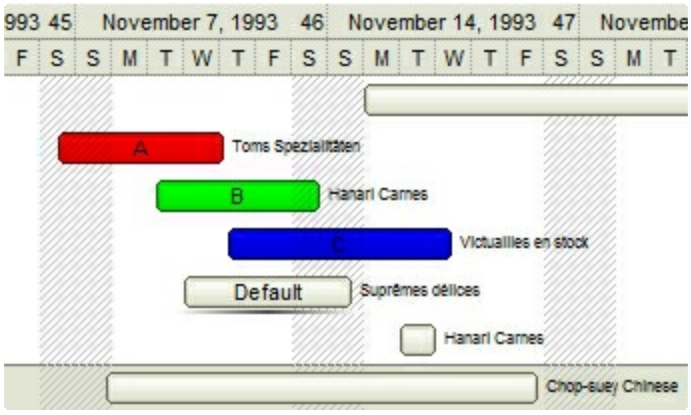
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



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

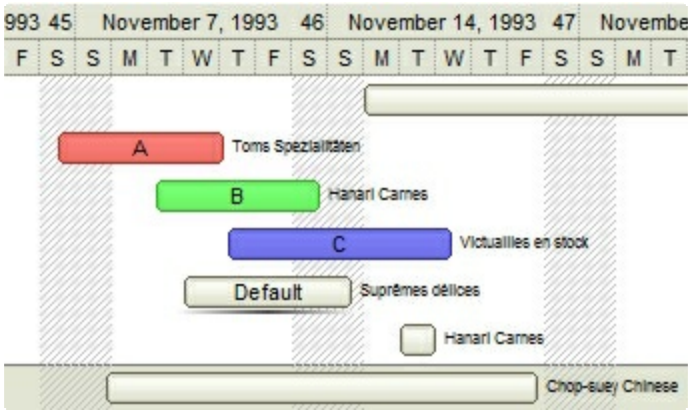


- ***-1**, AND-color scheme*, The color to be applied on the part of the control is an AND bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the AND bit for the entire Blue channel, or in other words, it applies a more Blue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



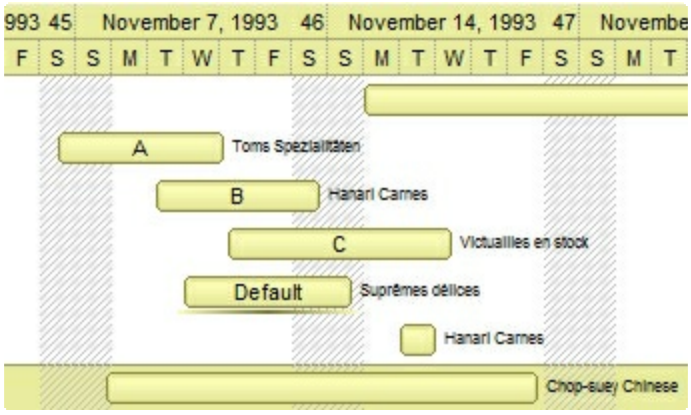
- ***0**, default*, the specified color is applied to the EBN. For instance, the

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

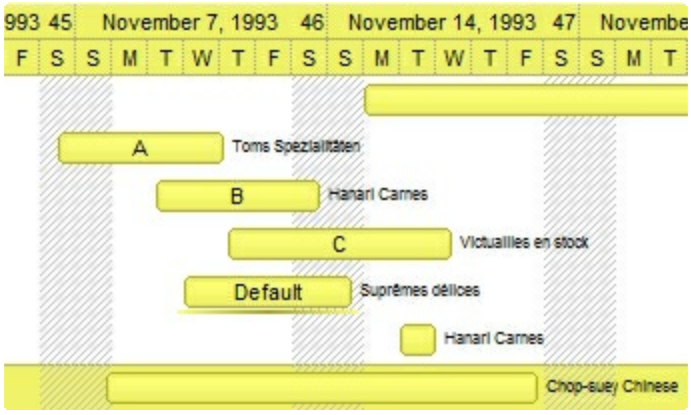


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

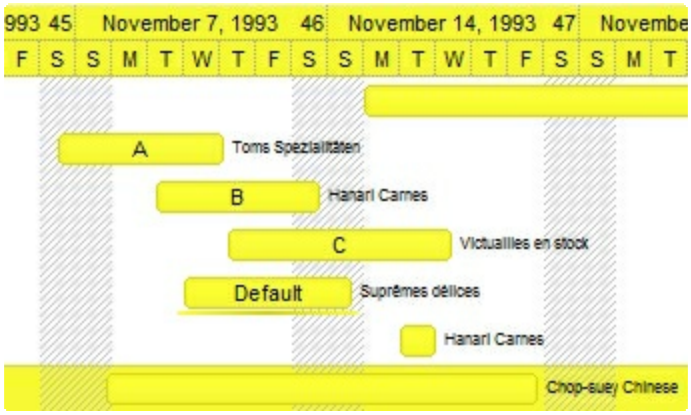
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



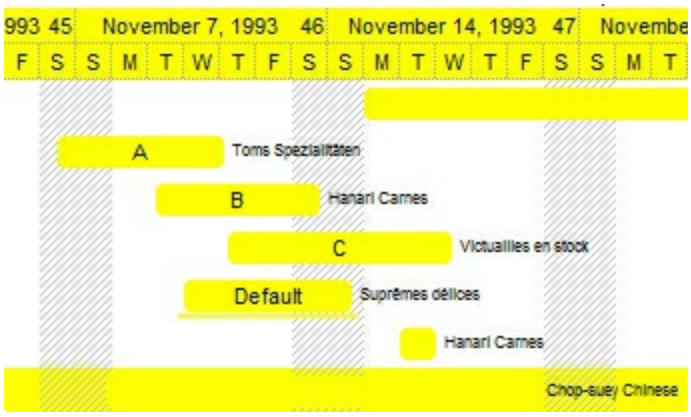
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



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



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



Item object

The Item object holds information about an item that's shown on the radial menu. Use the Add / ToString method to add new items to the control. Any item, can display a different HTML caption, icons, pictures on the items or sub-items zone.

The Item object supports the following properties and methods:

Name	Description
BackAlpha	Specifies the value of alpha / opacity channel to show the item's background color.
BackColor	Retrieves or sets a value that indicates the item's background color.
BrowseCustom	Gets or sets a value for specified property, when browsing custom data.
BrowseCustomType	Indicates the custom object to be shown when the user clicks/browses the item.
BrowseType	Specifies what the item displays, when the user clicks/browses it.
Caption	Retrieves or sets a value that indicates the item's caption.
ForeColor	Retrieves or sets a value that indicates the item's foreground color.
Image	Retrieves or sets a value that indicates the item's image.
Index	Retrieves the item's index.
Items	Retrieves the item's children collection.

Name	Retrieves or sets a value that indicates the item's name.
Parent	Retrieves the parent of the item.
Tooltip	Retrieves or sets a value that indicates the item's tooltip.
TooltipTitle	Retrieves or sets a value that indicates the title of the item's tooltip.
UserData	Retrieves or sets a value that indicates the item's user data.

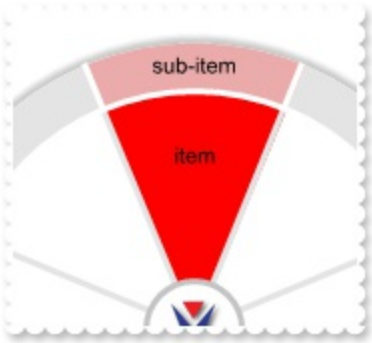
property Item.BackAlpha(Type as RadialItemsEnum) as Byte

Specifies the value of alpha / opacity channel to show the item's background color.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that indicates the part of the item to be changed.
Byte	A BYTE expression that specifies the value of alpha / opacity channel to show the item's background color. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

By default, the BackAlpha property is 255 (opaque). The BackAlpha property specifies the value of alpha / opacity channel to show the item's background color. The BackColor property retrieves or sets a value that indicates the item's background color.

The following screen shot shows an item with a different back color/alpha on item/sub-item portion:



The following samples show how you can change the item's back color/alpha:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .Expanded = True
   With .Items
      With .Add("item")
         .Caption(2) = "sub-item"
         .BackColor(3) = RGB(255,0,0)
         .BackAlpha(2) = 64
      End With
   End With
```

```
End With
```

VB6

```
With RadialMenu1
   .Expanded = True
   With .Items
      With .Add("item")
         .Caption(exRadialSubItems) = "sub-item"
         .BackColor(exRadialFullItems) = RGB(255,0,0)
         .BackAlpha(exRadialSubItems) = 64
      End With
   End With
End With
```

VB.NET

```
With Exradialmenu1
   .Expanded = True
   With .Items
      With .Add("item")

.set_Caption(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"sub-
item")

.set_BackColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,Color.F



.set_BackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,64)
      End With
   End With
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .Expanded = True
   With .Items

```

```
    With .Add("item")
      .Caption(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems) = "sub-item"
      .BackColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems) = RGB(255,0,0)
      .BackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems) = 64
    End With
  End With
End With
```

C++

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

   #import <ExRadialMenu.dll>
   using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 = GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
  EXRADIALMENULib::IItemPtr var_Item = var_Items->Add(L"item",vtMissing,vtMissing);
    var_Item->PutCaption(EXRADIALMENULib::exRadialSubItems,L"sub-item");
    var_Item->PutBackColor(EXRADIALMENULib::exRadialFullItems,RGB(255,0,0));
    var_Item->PutBackAlpha(EXRADIALMENULib::exRadialSubItems,64);
```

C++ Builder

```
RadialMenu1->Expanded = true;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
  Exradialmenulib_tlb::IItemPtr var_Item = var_Items->Add(L"item",TNoParam(),TNoParam());
    var_Item-
```

```
>set_Caption(Exradialmenulib_tlb::RadialItemsEnum::exRadialSubItems,L"sub-item");
      var_Item-
>set_BackColor(Exradialmenulib_tlb::RadialItemsEnum::exRadialFullItems,RGB(255,0,0))

      var_Item-
>set_BackAlpha(Exradialmenulib_tlb::RadialItemsEnum::exRadialSubItems,64);
```

C#

```
exradialmenu1.Expanded = true;
exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Item var_Item = var_Items.Add("item",null,null);

var_Item.set_Caption(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,
item");

var_Item.set_BackColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItem



var_Item.set_BackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubIten
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.Expanded = true;
   var var_Items = RadialMenu1.Items;
      var var_Item = var_Items.Add("item",null,null);
         var_Item.Caption(2) = "sub-item";
         var_Item.BackColor(3) = 255;
```

```
        var_Item.BackAlpha(2) = 64;
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .Expanded = True
      With .Items
         With .Add("item")
            .Caption(2) = "sub-item"
            .BackColor(3) = RGB(255,0,0)
            .BackAlpha(2) = 64
         End With
      End With
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.Expanded = true;
EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
   EXRADIALMENULib.Item var_Item = var_Items.Add("item",null,null);

var_Item.set_Caption(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"sub-
item");
      var_Item.set_BackColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,
```

```
(uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0)));

var_Item.set_BackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,64);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Item,com_Items;
    anytype var_Item,var_Items;
    ;

    super();

    exradialmenu1.Expanded(true);
    var_Items = exradialmenu1.Items(); com_Items = var_Items;
        var_Item = com_Items.Add("item"); com_Item = var_Item;
            com_Item.Caption(2/*exRadialSubItems*/,"sub-item");
            com_Item.BackColor(3/*exRadialFullItems*/,WinApi::RGB2int(255,0,0));
            com_Item.BackAlpha(2/*exRadialSubItems*/,64);
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
    Expanded := True;
    with Items do
    begin
        with Add('item',Nil,Nil) do
        begin
            Caption[EXRADIALMENULib.RadialItemsEnum.exRadialSubItems] := 'sub-item';
            BackColor[EXRADIALMENULib.RadialItemsEnum.exRadialFullItems] := $ff;
            BackAlpha[EXRADIALMENULib.RadialItemsEnum.exRadialSubItems] := 64;
        end;
    end;
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   Expanded := True;
   with Items do
   begin
      with Add('item',Null,Null) do
      begin
         Caption[EXRADIALMENULib_TLB.exRadialSubItems] := 'sub-item';
         BackColor[EXRADIALMENULib_TLB.exRadialFullItems] := $ff;
         BackAlpha[EXRADIALMENULib_TLB.exRadialSubItems] := 64;
      end;
   end;
end
```

VFP

```
with thisform.RadialMenu1
   .Expanded = .T.
   with .Items
      with .Add("item")
         .Caption(2) = "sub-item"
         .BackColor(3) = RGB(255,0,0)
         .BackAlpha(2) = 64
      endwith
   endwith
endwith
```

dBASE Plus

```
local oRadialMenu,var_Item,var_Items

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("item")
      // var_Item.Caption(2) = "sub-item"
```

```
    with (oRadialMenu)
      TemplateDef = [dim var_Item]
      TemplateDef = var_Item
      Template = [var_Item.Caption(2) = "sub-item"]
    endwith
    // var_Item.BackColor(3) = 0xff
    with (oRadialMenu)
      TemplateDef = [dim var_Item]
      TemplateDef = var_Item
      Template = [var_Item.BackColor(3) = 255]
    endwith
    // var_Item.BackAlpha(2) = 64
    with (oRadialMenu)
      TemplateDef = [dim var_Item]
      TemplateDef = var_Item
      Template = [var_Item.BackAlpha(2) = 64]
    endwith
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Item as P
Dim var_Items as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.Expanded = .t.
var_Items = oRadialMenu.Items
  var_Item = var_Items.Add("item")
    ' var_Item.Caption(2) = "sub-item"
    oRadialMenu.TemplateDef = "dim var_Item"
    oRadialMenu.TemplateDef = var_Item
    oRadialMenu.Template = "var_Item.Caption(2) = `sub-item`"

    ' var_Item.BackColor(3) = 255
    oRadialMenu.TemplateDef = "dim var_Item"
    oRadialMenu.TemplateDef = var_Item
```

```
    oRadialMenu.Template = "var_Item.BackColor(3) = 255"

    ' var_Item.BackAlpha(2) = 64
    oRadialMenu.TemplateDef = "dim var_Item"
    oRadialMenu.TemplateDef = var_Item
    oRadialMenu.Template = "var_Item.BackAlpha(2) = 64"
```

Visual Objects

```
local var_Item as IItem
local var_Items as IItems

oDCOCX_Exontrol1:Expanded := true
var_Items := oDCOCX_Exontrol1:Items
   var_Item := var_Items:Add("item",nil,nil)
      var_Item:[Caption,exRadialSubItems] := "sub-item"
      var_Item:[BackColor,exRadialFullItems] := RGB(255,0,0)
      var_Item:[BackAlpha,exRadialSubItems] := 64
```

PowerBuilder

```
OleObject oRadialMenu,var_Item,var_Items

oRadialMenu = ole_1.Object
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("item")
      var_Item.Caption(2,"sub-item")
      var_Item.BackColor(3,RGB(255,0,0))
      var_Item.BackAlpha(2,64)
```

Visual DataFlex

```
Procedure OnCreate
```

```
    Forward Send OnCreate
    Set ComExpanded to True
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Variant voItem
        Get ComAdd of hoItems "item" Nothing Nothing to voItem
        Handle hoItem
        Get Create (RefClass(cComItem)) to hoItem
        Set pvComObject of hoItem to voItem
            Set ComCaption of hoItem OLEexRadialSubItems to "sub-item"
            Set ComBackColor of hoItem OLEexRadialFullItems to (RGB(255,0,0))
            Set ComBackAlpha of hoItem OLEexRadialSubItems to 64
        Send Destroy to hoItem
    Send Destroy to hoItems
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItem
   LOCAL oItems
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
```

```
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:Expanded := .T.
      oItems := oRadialMenu:Items()
         oItem := oItems:Add("item")
            oItem:SetProperty("Caption",2/*exRadialSubItems*/,"sub-item")

oItem:SetProperty("BackColor",3/*exRadialFullItems*/,AutomationTranslateColor(
GraMakeRGBColor  ( { 255,0,0 } )  , .F. ))
            oItem:SetProperty("BackAlpha",2/*exRadialSubItems*/,64)

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

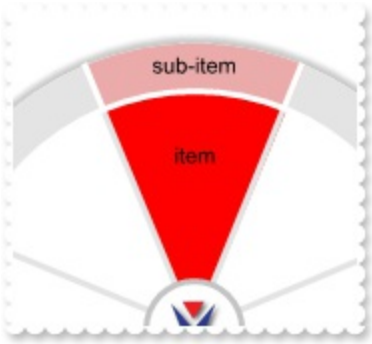
property Item.BackColor(Type as RadialItemsEnum) as Color

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

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that indicates the part of the item to be changed.
Color	A Color expression that defines the color to be applied. If -1 no color is applied.

By default, the BackColor property is -1, which indicates that no background color is applied to the item. The BackColor property retrieves or sets a value that indicates the item's background color. The BackAlpha property specifies the value of alpha / opacity channel to show the item's background color.

The following screen shot shows an item with a different back color/alpha on item/sub-item portion:



The following samples show how you can change the item's back color/alpha:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .Expanded = True
   With .Items
      With .Add("item")
         .Caption(2) = "sub-item"
         .BackColor(3) = RGB(255,0,0)
         .BackAlpha(2) = 64
      End With
   End With
End With
```

VB6

```
With RadialMenu1
   .Expanded = True
   With .Items
      With .Add("item")
         .Caption(exRadialSubItems) = "sub-item"
         .BackColor(exRadialFullItems) = RGB(255,0,0)
         .BackAlpha(exRadialSubItems) = 64
      End With
   End With
End With
```

VB.NET

```
With Exradialmenu1
   .Expanded = True
   With .Items
      With .Add("item")

.set_Caption(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"sub-
item")

.set_BackColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,Color.F



.set_BackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,64)
      End With
   End With
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .Expanded = True
   With .Items
      With .Add("item")
         .Caption(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems) = "sub-
```

```
item"
        .BackColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems) =
RGB(255,0,0)
        .BackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems) = 64
      End With
   End With
End With
```

C++

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

   #import <ExRadialMenu.dll>
   using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   EXRADIALMENULib::IItemPtr var_Item = var_Items-
>Add(L"item",vtMissing,vtMissing);
      var_Item->PutCaption(EXRADIALMENULib::exRadialSubItems,L"sub-item");
      var_Item->PutBackColor(EXRADIALMENULib::exRadialFullItems,RGB(255,0,0));
      var_Item->PutBackAlpha(EXRADIALMENULib::exRadialSubItems,64);
```

C++ Builder

```
RadialMenu1->Expanded = true;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   Exradialmenulib_tlb::IItemPtr var_Item = var_Items-
>Add(L"item",TNoParam(),TNoParam());
      var_Item-
>set_Caption(Exradialmenulib_tlb::RadialItemsEnum::exRadialSubItems,L"sub-item");
      var_Item-
```

```
>set_BackColor(ExradialmenuLib_tlb::RadialItemsEnum::exRadialFullItems,RGB(255,0,0))

      var_Item-
>set_BackAlpha(ExradialmenuLib_tlb::RadialItemsEnum::exRadialSubItems,64);
```

C#

```
exradialmenu1.Expanded = true;
exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Item var_Item = var_Items.Add("item",null,null);

var_Item.set_Caption(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,
item");

var_Item.set_BackColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItem



var_Item.set_BackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubIten
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.Expanded = true;
   var var_Items = RadialMenu1.Items;
      var var_Item = var_Items.Add("item",null,null);
         var_Item.Caption(2) = "sub-item";
         var_Item.BackColor(3) = 255;
         var_Item.BackAlpha(2) = 64;
}
```

```
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .Expanded = True
      With .Items
         With .Add("item")
            .Caption(2) = "sub-item"
            .BackColor(3) = RGB(255,0,0)
            .BackAlpha(2) = 64
         End With
      End With
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.Expanded = true;
EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
   EXRADIALMENULib.Item var_Item = var_Items.Add("item",null,null);

var_Item.set_Caption(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"sub-
item");
      var_Item.set_BackColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,
(uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0)));
```

```
var_Item.set_BackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,64);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Item,com_Items;
    anytype var_Item,var_Items;
    ;

    super();

    exradialmenu1.Expanded(true);
    var_Items = exradialmenu1.Items(); com_Items = var_Items;
        var_Item = com_Items.Add("item"); com_Item = var_Item;
            com_Item.Caption(2/*exRadialSubItems*/,"sub-item");
            com_Item.BackColor(3/*exRadialFullItems*/,WinApi::RGB2int(255,0,0));
            com_Item.BackAlpha(2/*exRadialSubItems*/,64);
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
    Expanded := True;
    with Items do
    begin
        with Add('item',Nil,Nil) do
        begin
            Caption[EXRADIALMENULib.RadialItemsEnum.exRadialSubItems] := 'sub-item';
            BackColor[EXRADIALMENULib.RadialItemsEnum.exRadialFullItems] := $ff;
            BackAlpha[EXRADIALMENULib.RadialItemsEnum.exRadialSubItems] := 64;
        end;
    end;
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   Expanded := True;
   with Items do
   begin
      with Add('item',Null,Null) do
      begin
         Caption[EXRADIALMENULib_TLB.exRadialSubItems] := 'sub-item';
         BackColor[EXRADIALMENULib_TLB.exRadialFullItems] := $ff;
         BackAlpha[EXRADIALMENULib_TLB.exRadialSubItems] := 64;
      end;
   end;
end
```

VFP

```
with thisform.RadialMenu1
   .Expanded = .T.
   with .Items
      with .Add("item")
         .Caption(2) = "sub-item"
         .BackColor(3) = RGB(255,0,0)
         .BackAlpha(2) = 64
      endwith
   endwith
endwith
```

dBASE Plus

```
local oRadialMenu,var_Item,var_Items

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("item")
      // var_Item.Caption(2) = "sub-item"
      with (oRadialMenu)
         TemplateDef = [dim var_Item]
```

```
      TemplateDef = var_Item
      Template = [var_Item.Caption(2) = "sub-item"]
   endwith
   // var_Item.BackColor(3) = 0xff
   with (oRadialMenu)
      TemplateDef = [dim var_Item]
      TemplateDef = var_Item
      Template = [var_Item.BackColor(3) = 255]
   endwith
   // var_Item.BackAlpha(2) = 64
   with (oRadialMenu)
      TemplateDef = [dim var_Item]
      TemplateDef = var_Item
      Template = [var_Item.BackAlpha(2) = 64]
   endwith
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Item as P
Dim var_Items as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.Expanded = .t.
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("item")
      ' var_Item.Caption(2) = "sub-item"
      oRadialMenu.TemplateDef = "dim var_Item"
      oRadialMenu.TemplateDef = var_Item
      oRadialMenu.Template = "var_Item.Caption(2) = `sub-item`"

      ' var_Item.BackColor(3) = 255
      oRadialMenu.TemplateDef = "dim var_Item"
      oRadialMenu.TemplateDef = var_Item
      oRadialMenu.Template = "var_Item.BackColor(3) = 255"
```

```
    ' var_Item.BackAlpha(2) = 64
    oRadialMenu.TemplateDef = "dim var_Item"
    oRadialMenu.TemplateDef = var_Item
    oRadialMenu.Template = "var_Item.BackAlpha(2) = 64"
```

Visual Objects

```
local var_Item as IItem
local var_Items as IItems

oDCOCX_Exontrol1:Expanded := true
var_Items := oDCOCX_Exontrol1:Items
   var_Item := var_Items:Add("item",nil,nil)
      var_Item:[Caption,exRadialSubItems] := "sub-item"
      var_Item:[BackColor,exRadialFullItems] := RGB(255,0,0)
      var_Item:[BackAlpha,exRadialSubItems] := 64
```

PowerBuilder

```
OleObject oRadialMenu,var_Item,var_Items

oRadialMenu = ole_1.Object
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("item")
      var_Item.Caption(2,"sub-item")
      var_Item.BackColor(3,RGB(255,0,0))
      var_Item.BackAlpha(2,64)
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Set ComExpanded to True
```

```
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Variant voItem
        Get ComAdd of hoItems "item" Nothing Nothing to voItem
        Handle hoItem
        Get Create (RefClass(cComItem)) to hoItem
        Set pvComObject of hoItem to voItem
            Set ComCaption of hoItem OLEexRadialSubItems to "sub-item"
            Set ComBackColor of hoItem OLEexRadialFullItems to (RGB(255,0,0))
            Set ComBackAlpha of hoItem OLEexRadialSubItems to 64
        Send Destroy to hoItem
    Send Destroy to hoItems
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItem
   LOCAL oItems
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
```

```
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:Expanded := .T.
      oItems := oRadialMenu:Items()
         oItem := oItems:Add("item")
            oItem:SetProperty("Caption",2/*exRadialSubItems*/,"sub-item")

oItem:SetProperty("BackColor",3/*exRadialFullItems*/,AutomationTranslateColor(
GraMakeRGBColor  ( { 255,0,0 } )  , .F. ))
            oItem:SetProperty("BackAlpha",2/*exRadialSubItems*/,64)

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

property Item.BrowseCustom(Property as RadialCustomPropertyEnum) as Variant

Gets or sets a value for specified property, when browsing custom data.

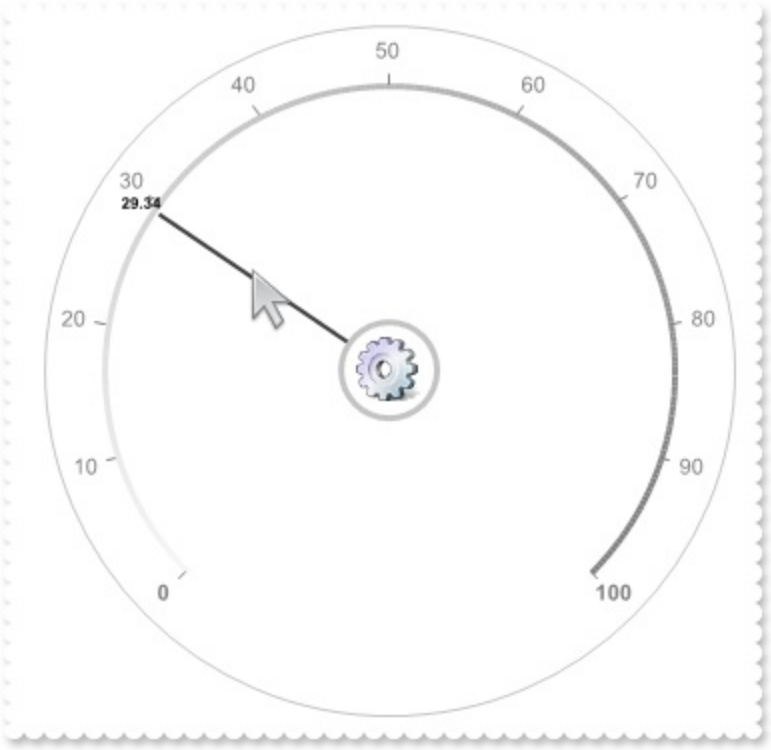
Type	Description
Property as RadialCustomPropertyEnum	A RadialCustomPropertyEnum expression that specifies the custom property to be changed.
Variant	A VARIANT expression that indicates the value of the giving property.

The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item. The BrowseCustom property should be used when the BrowseType property is exBrowseItemCustom.

When an item is browsing, it can display any of the following:

- child items (BrowseType property is exBrowseItemChild)

- radial-slider (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomSlider)



- gauge control (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomGauge). The control displays/edit data using the using the Exontrol's ExGauge component.

The following sample, shows how you can add a gauge control:

```
With RadialMenu1
```

```
    .BeginUpdate
    .Expanded = True
    .InflateCustom = "-16 * dpi"
    .Caption(exLayerCaption) = .FormatABC("`ExGauge Version: ` + value", Gauge1.Version)
    With .Items.Add("<c>ExGauge<br><c>Inside<br><c>Clock")
       .BrowseType = exBrowseItemCustom
       .BrowseCustomType = exRadialCustomGauge
       .BrowseCustom(exRadialCustomGaugeHandle) = Gauge1.hWnd
    End With
    .EndUpdate
End With
```

property Item.BrowseCustomType as RadialCustomTypeEnum

Indicates the custom object to be shown when the user clicks/browses the item.

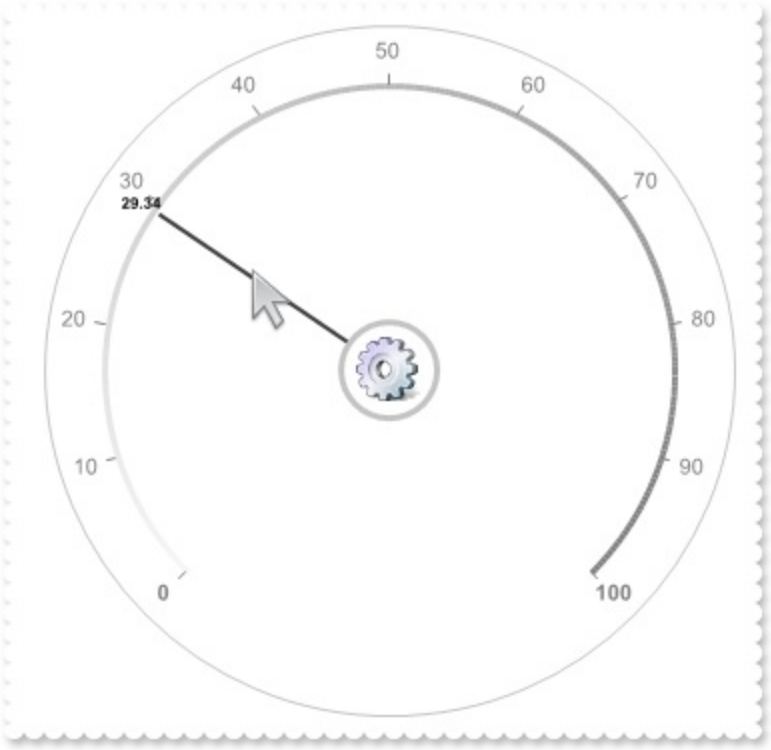
Type	Description
RadialCustomTypeEnum	A RadialCustomTypeEnum expression that specifies the custom control to be displayed when user browses the item.

By default, the BrowseCustomType property is exRadialCustomDisable, which indicates that the control displays no custom control, once the user clicks / selects / browses the item. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item. The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property must be specified when the BrowseType property is exBrowseItemCustom.

When an item is browsing, it can display any of the following:

- child items (BrowseType property is exBrowseItemChild)

- radial-slider (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomSlider)



- gauge control (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomGauge). The control displays/edit data using the using the Exontrol's ExGauge component.

The following samples show how you can add a radial-slider:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .Expanded = True
   With .Items
      With .Add("Slider")
         .BrowseType = 2
         .BrowseCustomType = 16
         .BrowseCustom(9) = 35
      End With
   End With
   .BrowseItem = .Items.Item("Slider")
End With
```

VB6

```
With RadialMenu1
   .Expanded = True
   With .Items
      With .Add("Slider")
         .BrowseType = exBrowseItemCustom
         .BrowseCustomType = exRadialCustomSlider
         .BrowseCustom(exRadialCustomSliderValue) = 35
      End With
   End With
   .BrowseItem = .Items.Item("Slider")
End With
```

VB.NET

```
With Exradialmenu1
   .Expanded = True
   With .Items
      With .Add("Slider")
         .BrowseType =
exontrol.EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom
         .BrowseCustomType =
exontrol.EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider

.set_BrowseCustom(exontrol.EXRADIALMENULib.RadialCustomPropertyEnum.exRadial
```

```
      End With
   End With
   .BrowseItem = .Items.Item("Slider")
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .Expanded = True
   With .Items
      With .Add("Slider")
         .BrowseType = EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom
         .BrowseCustomType =
EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider

.BrowseCustom(EXRADIALMENULib.RadialCustomPropertyEnum.exRadialCustomSlider
 = 35
      End With
   End With
   .BrowseItem = .Items.Item("Slider")
End With
```

C++

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

   #import <ExRadialMenu.dll>
   using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   EXRADIALMENULib::IItemPtr var_Item = var_Items-
```

```
>Add(L"Slider",vtMissing,vtMissing);
      var_Item->PutBrowseType(EXRADIALMENULib::exBrowseItemCustom);
      var_Item->PutBrowseCustomType(EXRADIALMENULib::exRadialCustomSlider);
      var_Item-
>PutBrowseCustom(EXRADIALMENULib::exRadialCustomSliderValue,long(35));
spRadialMenu1->PutBrowseItem(((EXRADIALMENULib::IItemPtr)(spRadialMenu1-
>GetItems()->GetItem("Slider"))));
```

C++ Builder

```
RadialMenu1->Expanded = true;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   Exradialmenulib_tlb::IItemPtr var_Item = var_Items-
>Add(L"Slider",TNoParam(),TNoParam());
      var_Item->BrowseType =
Exradialmenulib_tlb::BrowseItemEnum::exBrowseItemCustom;
      var_Item->BrowseCustomType =
Exradialmenulib_tlb::RadialCustomTypeEnum::exRadialCustomSlider;
      var_Item-
>set_BrowseCustom(Exradialmenulib_tlb::RadialCustomPropertyEnum::exRadialCustom

RadialMenu1->BrowseItem = RadialMenu1->Items->get_Item(TVariant("Slider"));
```

C#

```
exradialmenu1.Expanded = true;
exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Item var_Item = var_Items.Add("Slider",null,null);
      var_Item.BrowseType =
exontrol.EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom;
      var_Item.BrowseCustomType =
exontrol.EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider;

var_Item.set_BrowseCustom(exontrol.EXRADIALMENULib.RadialCustomPropertyEnum.

exradialmenu1.BrowseItem = (exradialmenu1.Items["Slider"] as
```

```
exontrol.EXRADIALMENULib.Item);
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.Expanded = true;
   var var_Items = RadialMenu1.Items;
      var var_Item = var_Items.Add("Slider",null,null);
         var_Item.BrowseType = 2;
         var_Item.BrowseCustomType = 16;
         var_Item.BrowseCustom(9) = 35;
   RadialMenu1.BrowseItem = RadialMenu1.Items.Item("Slider");
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .Expanded = True
      With .Items
         With .Add("Slider")
            .BrowseType = 2
            .BrowseCustomType = 16
```

```
            .BrowseCustom(9) = 35
          End With
        End With
        .BrowseItem = .Items.Item("Slider")
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.Expanded = true;
EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
    EXRADIALMENULib.Item var_Item = var_Items.Add("Slider",null,null);
        var_Item.BrowseType =
EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom;
        var_Item.BrowseCustomType =
EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider;

var_Item.set_BrowseCustom(EXRADIALMENULib.RadialCustomPropertyEnum.exRadialC

axRadialMenu1.BrowseItem = (axRadialMenu1.Items["Slider"] as
EXRADIALMENULib.Item);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Item,com_Items;
    anytype var_Item,var_Items;
    ;

    super();

    exradialmenu1.Expanded(true);
    var_Items = exradialmenu1.Items(); com_Items = var_Items;
```

```
      var_Item = com_Items.Add("Slider"); com_Item = var_Item;
         com_Item.BrowseType(2/*exBrowseItemCustom*/);
         com_Item.BrowseCustomType(16/*exRadialCustomSlider*/);

com_Item.BrowseCustom(9/*exRadialCustomSliderValue*/,COMVariant::createFromInt(

   exradialmenu1.BrowseItem(exradialmenu1.Items().Item("Slider"));
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
   Expanded := True;
   with Items do
   begin
      with Add('Slider',Nil,Nil) do
      begin
         BrowseType := EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom;
         BrowseCustomType :=
EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider;

BrowseCustom[EXRADIALMENULib.RadialCustomPropertyEnum.exRadialCustomSlider
 := TObject(35);
      end;
   end;
   BrowseItem := (Items.Item['Slider'] as EXRADIALMENULib.Item);
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   Expanded := True;
   with Items do
   begin
      with Add('Slider',Null,Null) do
      begin
```

```
        BrowseType := EXRADIALMENULib_TLB.exBrowseItemCustom;
        BrowseCustomType := EXRADIALMENULib_TLB.exRadialCustomSlider;
        BrowseCustom[EXRADIALMENULib_TLB.exRadialCustomSliderValue] :=
OleVariant(35);
      end;
   end;
   BrowseItem := (IUnknown(Items.Item['Slider']) as EXRADIALMENULib_TLB.Item);
end
```

VFP

```
with thisform.RadialMenu1
   .Expanded = .T.
   with .Items
      with .Add("Slider")
         .BrowseType = 2
         .BrowseCustomType = 16
         .BrowseCustom(9) = 35
      endwith
   endwith
   .BrowseItem = .Items.Item("Slider")
endwith
```

dBASE Plus

```
local oRadialMenu,var_Item,var_Items

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("Slider")
      var_Item.BrowseType = 2
      var_Item.BrowseCustomType = 16
      // var_Item.BrowseCustom(9) = 35
      with (oRadialMenu)
         TemplateDef = [dim var_Item]
         TemplateDef = var_Item
         Template = [var_Item.BrowseCustom(9) = 35]
```

```
    endwith
oRadialMenu.BrowseItem = oRadialMenu.Items.Item("Slider")
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Item as P
Dim var_Items as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.Expanded = .t.
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("Slider")
      var_Item.BrowseType = 2
      var_Item.BrowseCustomType = 16
      ' var_Item.BrowseCustom(9) = 35
      oRadialMenu.TemplateDef = "dim var_Item"
      oRadialMenu.TemplateDef = var_Item
      oRadialMenu.Template = "var_Item.BrowseCustom(9) = 35"

oRadialMenu.BrowseItem = oRadialMenu.Items.Item("Slider")
```

Visual Objects

```
local var_Item as IItem
local var_Items as IItems

oDCOCX_Exontrol1:Expanded := true
var_Items := oDCOCX_Exontrol1:Items
   var_Item := var_Items:Add("Slider",nil,nil)
      var_Item:BrowseType := exBrowseItemCustom
      var_Item:BrowseCustomType := exRadialCustomSlider
      var_Item:[BrowseCustom,exRadialCustomSliderValue] := 35
oDCOCX_Exontrol1:BrowseItem := IItem{oDCOCX_Exontrol1:Items:[Item,"Slider"]}
```

PowerBuilder

```
OleObject oRadialMenu,var_Item,var_Items

oRadialMenu = ole_1.Object
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("Slider")
      var_Item.BrowseType = 2
      var_Item.BrowseCustomType = 16
      var_Item.BrowseCustom(9,35)
oRadialMenu.BrowseItem = oRadialMenu.Items.Item("Slider")
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Set ComExpanded to True
   Variant voItems
   Get ComItems to voItems
   Handle hoItems
   Get Create (RefClass(cComItems)) to hoItems
   Set pvComObject of hoItems to voItems
      Variant voItem
      Get ComAdd of hoItems "Slider" Nothing Nothing to voItem
      Handle hoItem
      Get Create (RefClass(cComItem)) to hoItem
      Set pvComObject of hoItem to voItem
         Set ComBrowseType of hoItem to OLEexBrowseItemCustom
         Set ComBrowseCustomType of hoItem to OLEexRadialCustomSlider
         Set ComBrowseCustom of hoItem OLEexRadialCustomSliderValue to 35
      Send Destroy to hoItem
   Send Destroy to hoItems
   Variant v
   Variant voItems1
   Get ComItems to voItems1
   Handle hoItems1
```

```
    Get Create (RefClass(cComItems)) to hoItems1
    Set pvComObject of hoItems1 to voItems1
        Get ComItem of hoItems1 "Slider" to v
    Send Destroy to hoItems1
    Set ComBrowseItem to v
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItem
   LOCAL oItems
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:Expanded := .T.
      oItems := oRadialMenu:Items()
         oItem := oItems:Add("Slider")
            oItem:BrowseType := 2/*exBrowseItemCustom*/
            oItem:BrowseCustomType := 16/*exRadialCustomSlider*/
            oItem:SetProperty("BrowseCustom",9/*exRadialCustomSliderValue*/,35)
      oRadialMenu:BrowseItem := oRadialMenu:Items:Item("Slider")
```

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

property Item.BrowseType as BrowseItemEnum

Specifies what the item displays, when the user clicks/browses it.

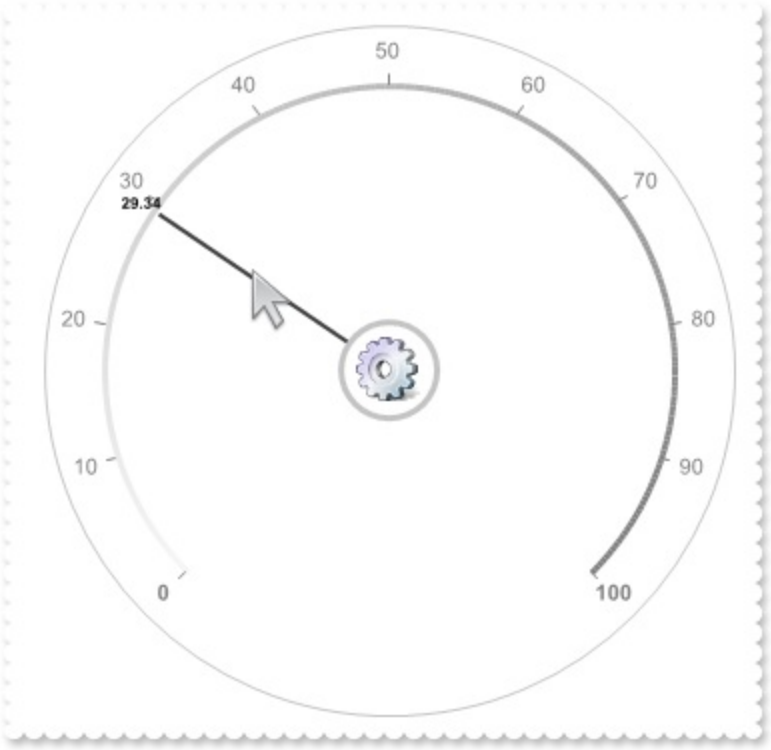
Type	Description
BrowseItemEnum	A BrowseItemEnum expression that specifies what the item displays, when the user clicks/browses it.

By default, the BrowseType property is exBrowseItemChild, which indicates that the control displays the child items, once the user clicks / selects / browses the item. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item. The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

When an item is browsing, it can display any of the following:

- child items (BrowseType property is exBrowseItemChild)

- radial-slider (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomSlider)



- gauge control (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomGauge). The control displays/edit data using the using the Exontrol's ExGauge component.

The following samples show how you can add new items / children to the control :

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   With .Items
      With .Add("Item 1").Items
         .Add "SubItem 1"
         .Add "SubItem 2"
      End With
      .Add "Item 2"
      .Add "Item 3"
      .Add "Item 4"
      .Add "Item 5"
      .Add "Item 6"
      .Add "Item 7"
      .Add "Item 8"
   End With
   .EndUpdate
End With
```

VB6

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   With .Items
      With .Add("Item 1").Items
         .Add "SubItem 1"
         .Add "SubItem 2"
      End With
      .Add "Item 2"
      .Add "Item 3"
      .Add "Item 4"
      .Add "Item 5"
      .Add "Item 6"
      .Add "Item 7"
      .Add "Item 8"
   End With
```

```
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
    .Expanded = True
    With .Items
        With .Add("Item 1").Items
            .Add("SubItem 1")
            .Add("SubItem 2")
        End With
        .Add("Item 2")
        .Add("Item 3")
        .Add("Item 4")
        .Add("Item 5")
        .Add("Item 6")
        .Add("Item 7")
        .Add("Item 8")
    End With
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
    .BeginUpdate()
    .Expanded = True
    With .Items
        With .Add("Item 1").Items
            .Add("SubItem 1")
            .Add("SubItem 2")
        End With
        .Add("Item 2")
        .Add("Item 3")
        .Add("Item 4")
        .Add("Item 5")
```

```
      .Add("Item 6")
      .Add("Item 7")
      .Add("Item 8")
   End With
   .EndUpdate()
End With
```

C++

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

   #import <ExRadialMenu.dll>
   using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   EXRADIALMENULib::IItemsPtr var_Items1 = var_Items->Add(L"Item
1",vtMissing,vtMissing)->GetItems();
      var_Items1->Add(L"SubItem 1",vtMissing,vtMissing);
      var_Items1->Add(L"SubItem 2",vtMissing,vtMissing);
   var_Items->Add(L"Item 2",vtMissing,vtMissing);
   var_Items->Add(L"Item 3",vtMissing,vtMissing);
   var_Items->Add(L"Item 4",vtMissing,vtMissing);
   var_Items->Add(L"Item 5",vtMissing,vtMissing);
   var_Items->Add(L"Item 6",vtMissing,vtMissing);
   var_Items->Add(L"Item 7",vtMissing,vtMissing);
   var_Items->Add(L"Item 8",vtMissing,vtMissing);
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   Exradialmenulib_tlb::IItemsPtr var_Items1 = var_Items->Add(L"Item
1",TNoParam(),TNoParam())->Items;
      var_Items1->Add(L"SubItem 1",TNoParam(),TNoParam());
      var_Items1->Add(L"SubItem 2",TNoParam(),TNoParam());
   var_Items->Add(L"Item 2",TNoParam(),TNoParam());
   var_Items->Add(L"Item 3",TNoParam(),TNoParam());
   var_Items->Add(L"Item 4",TNoParam(),TNoParam());
   var_Items->Add(L"Item 5",TNoParam(),TNoParam());
   var_Items->Add(L"Item 6",TNoParam(),TNoParam());
   var_Items->Add(L"Item 7",TNoParam(),TNoParam());
   var_Items->Add(L"Item 8",TNoParam(),TNoParam());
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Items var_Items1 = var_Items.Add("Item
1",null,null).Items;
      var_Items1.Add("SubItem 1",null,null);
      var_Items1.Add("SubItem 2",null,null);
   var_Items.Add("Item 2",null,null);
   var_Items.Add("Item 3",null,null);
   var_Items.Add("Item 4",null,null);
   var_Items.Add("Item 5",null,null);
   var_Items.Add("Item 6",null,null);
   var_Items.Add("Item 7",null,null);
   var_Items.Add("Item 8",null,null);
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.Expanded = true;
   var var_Items = RadialMenu1.Items;
      var var_Items1 = var_Items.Add("Item 1",null,null).Items;
         var_Items1.Add("SubItem 1",null,null);
         var_Items1.Add("SubItem 2",null,null);
      var_Items.Add("Item 2",null,null);
      var_Items.Add("Item 3",null,null);
      var_Items.Add("Item 4",null,null);
      var_Items.Add("Item 5",null,null);
      var_Items.Add("Item 6",null,null);
      var_Items.Add("Item 7",null,null);
      var_Items.Add("Item 8",null,null);
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .Expanded = True
```

```
    With .Items
      With .Add("Item 1").Items
        .Add "SubItem 1"
        .Add "SubItem 2"
      End With
      .Add "Item 2"
      .Add "Item 3"
      .Add "Item 4"
      .Add "Item 5"
      .Add "Item 6"
      .Add "Item 7"
      .Add "Item 8"
    End With
    .EndUpdate
  End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
  EXRADIALMENULib.Items var_Items1 = var_Items.Add("Item 1",null,null).Items;
    var_Items1.Add("SubItem 1",null,null);
    var_Items1.Add("SubItem 2",null,null);
  var_Items.Add("Item 2",null,null);
  var_Items.Add("Item 3",null,null);
  var_Items.Add("Item 4",null,null);
  var_Items.Add("Item 5",null,null);
  var_Items.Add("Item 6",null,null);
  var_Items.Add("Item 7",null,null);
  var_Items.Add("Item 8",null,null);
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Item,com_Items,com_Items1;
    anytype var_Item,var_Items,var_Items1;
    ;

    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.Expanded(true);
    var_Items = exradialmenu1.Items(); com_Items = var_Items;
        var_Item = COM::createFromObject(com_Items.Add("Item 1")); com_Item =
var_Item;
        var_Items1 = com_Item.Items(); com_Items1 = var_Items1;
            com_Items1.Add("SubItem 1");
            com_Items1.Add("SubItem 2");
        com_Items.Add("Item 2");
        com_Items.Add("Item 3");
        com_Items.Add("Item 4");
        com_Items.Add("Item 5");
        com_Items.Add("Item 6");
        com_Items.Add("Item 7");
        com_Items.Add("Item 8");
    exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
    BeginUpdate();
    Expanded := True;
    with Items do
    begin
        with Add('Item 1',Nil,Nil).Items do
        begin
```

```
        Add('SubItem 1',Nil,Nil);
        Add('SubItem 2',Nil,Nil);
      end;
      Add('Item 2',Nil,Nil);
      Add('Item 3',Nil,Nil);
      Add('Item 4',Nil,Nil);
      Add('Item 5',Nil,Nil);
      Add('Item 6',Nil,Nil);
      Add('Item 7',Nil,Nil);
      Add('Item 8',Nil,Nil);
    end;
    EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
    BeginUpdate();
    Expanded := True;
    with Items do
    begin
      with Add('Item 1',Null,Null).Items do
      begin
        Add('SubItem 1',Null,Null);
        Add('SubItem 2',Null,Null);
      end;
      Add('Item 2',Null,Null);
      Add('Item 3',Null,Null);
      Add('Item 4',Null,Null);
      Add('Item 5',Null,Null);
      Add('Item 6',Null,Null);
      Add('Item 7',Null,Null);
      Add('Item 8',Null,Null);
    end;
    EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
   .BeginUpdate
   .Expanded = .T.
   with .Items
      with .Add("Item 1").Items
         .Add("SubItem 1")
         .Add("SubItem 2")
      endwith
      .Add("Item 2")
      .Add("Item 3")
      .Add("Item 4")
      .Add("Item 5")
      .Add("Item 6")
      .Add("Item 7")
      .Add("Item 8")
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu,var_Items,var_Items1

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
```

```
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Items as P
Dim var_Items1 as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = .t.
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

Visual Objects

```
local var_Items,var_Items1 as IItems

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
var_Items := oDCOCX_Exontrol1:Items
   var_Items1 := var_Items:Add("Item 1",nil,nil):Items
      var_Items1:Add("SubItem 1",nil,nil)
      var_Items1:Add("SubItem 2",nil,nil)
   var_Items:Add("Item 2",nil,nil)
```

```
   var_Items:Add("Item 3",nil,nil)
   var_Items:Add("Item 4",nil,nil)
   var_Items:Add("Item 5",nil,nil)
   var_Items:Add("Item 6",nil,nil)
   var_Items:Add("Item 7",nil,nil)
   var_Items:Add("Item 8",nil,nil)
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu,var_Items,var_Items1

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComExpanded to True
   Variant voItems
   Get ComItems to voItems
```

```
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Variant voItem
        Get ComAdd of hoItems "Item 1" Nothing Nothing to voItem
        Handle hoItem
        Get Create (RefClass(cComItem)) to hoItem
        Set pvComObject of hoItem to voItem
            Variant voItems1
            Get ComItems of hoItem to voItems1
            Handle hoItems1
            Get Create (RefClass(cComItems)) to hoItems1
            Set pvComObject of hoItems1 to voItems1
                Get ComAdd of hoItems1 "SubItem 1" Nothing Nothing to Nothing
                Get ComAdd of hoItems1 "SubItem 2" Nothing Nothing to Nothing
            Send Destroy to hoItems1
        Send Destroy to hoItem
        Get ComAdd of hoItems "Item 2" Nothing Nothing to Nothing
        Get ComAdd of hoItems "Item 3" Nothing Nothing to Nothing
        Get ComAdd of hoItems "Item 4" Nothing Nothing to Nothing
        Get ComAdd of hoItems "Item 5" Nothing Nothing to Nothing
        Get ComAdd of hoItems "Item 6" Nothing Nothing to Nothing
        Get ComAdd of hoItems "Item 7" Nothing Nothing to Nothing
        Get ComAdd of hoItems "Item 8" Nothing Nothing to Nothing
    Send Destroy to hoItems
    Send ComEndUpdate
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItems,oItems1
```

```
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
      oItems := oRadialMenu:Items()
         oItems1 := oItems:Add("Item 1"):Items()
            oItems1:Add("SubItem 1")
            oItems1:Add("SubItem 2")
         oItems:Add("Item 2")
         oItems:Add("Item 3")
         oItems:Add("Item 4")
         oItems:Add("Item 5")
         oItems:Add("Item 6")
         oItems:Add("Item 7")
         oItems:Add("Item 8")
      oRadialMenu:EndUpdate()

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

property Item.Caption(Type as RadialItemsEnum) as String

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

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that indicates the part of the item to be changed.
String	A String expression that specifies the caption of the item. The Caption property supports HTML format as described bellow.

By default, the Caption property is empty. The Caption property retrieves or sets a value that indicates the item's caption. You can specify the caption of the item using the Caption parameter of the Add method. The ForeColor property specifies the item's foreground color. The Image property assigns an icon/picture to the item. The Name property of the Item object is equivalent with the Caption(exRadialItems) property. The UserData property retrieves or sets a value that indicates the item's user data. The DisplayCenter property specifies the ratio to determine where the image/caption of the item is displayed.

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

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

;12>bit" displays the bit text using the current font, but with a different size.

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

to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

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

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Item.ForeColor(Type as RadialItemsEnum) as Color

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

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that indicates the part of the item to be changed.
Color	A Color expression that specifies the item's foreground color. If -1, the control's ForeColor property indicates the item's foreground color.

By default, the ForeColor property is -1, which indicates that the control's ForeColor property specifies the item's foreground color. The ForeColor property specifies the item's foreground color. The Caption property retrieves or sets a value that indicates the item's caption. You can specify the caption of the item using the Caption parameter of the Add method. The Image property assigns an icon/picture to the item.

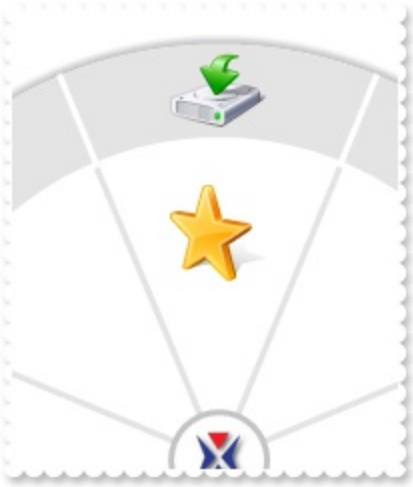
property Item.Image(Type as RadialItemsEnum) as Variant

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

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that indicates the part of the item to be changed.

Variant	A VARIANT expression that specifies the icon/picture/image to be displayed as described:
	
	• A String expression indicates:
	  ◦ a name of a picture file in the PicturePath folder. For instance, Image(exRadialItems) = "favorites.png", loads the favorites.png file if found in the PicturePath folder.
	  ◦ a picture file including its absolute path. For instance, Image(exRadialItems) = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favo loads the favorites.png file from absolute path
	  ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, Image(exRadialItems) = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favo
	  ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on.
	• A Picture object that indicates the picture to be displayed. For instance, Image(exRadialItems) = LoadPicture("picture.jpg")
	• a long/string expression that specifies the index of the icon to be displayed (0-based). The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection.
	
	If no icon/picture/image is found, the item displays no icon/picture/image.

By default, the Image property is empty, which indicates that no image is displayed. The Image property retrieves or sets a value that indicates the item's image. You can specify the image of the item using the Image parameter of the Add method. The Caption property retrieves or sets a value that indicates the item's caption. The item's caption may display any icon, picture or image, using the built-in HTML **<img>** tag. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ItemsImageWidth / ItemsImageHeight property specifies the size to display the item's image. The DisplayCenter property specifies the ratio to determine where the image/caption of the item is displayed.



The following samples show how you can display images within the control:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .Expanded = True
   .SubItemsSize = "48*dpi"
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .RadialLineColor(9) = RGB(128,128,128)
   .RadialLineColor(10) = RGB(128,128,128)
   .RadialLineColor(11) = -1
   With .Items
      With .Add("")
         .Image(1) = "favorites.png"
         .Image(2) = "download.png"
      End With
   End With
End With
```

VB6

```
With RadialMenu1
   .Expanded = True
   .SubItemsSize = "48*dpi"
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .RadialLineColor(exRadialHotItem) = RGB(128,128,128)
   .RadialLineColor(exRadialHotSubItem) = RGB(128,128,128)
   .RadialLineColor(exRadialHotFullItem) = -1
   With .Items
      With .Add("")
         .Image(exRadialItems) = "favorites.png"
         .Image(exRadialSubItems) = "download.png"
      End With
   End With
End With
```

VB.NET

```
With Exradialmenu1
   .Expanded = True
   .SubItemsSize = "48*dpi"
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"

.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotItem,Col



.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotSubItem



.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotFullIte

   With .Items
      With .Add("")

.set_Image(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,"favorites.png



.set_Image(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"downloa
```

```
        End With
    End With
End With
```

VB.NET for /COM

```
With AxRadialMenu1
    .Expanded = True
    .SubItemsSize = "48*dpi"
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"

.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotItem,8421504)

.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotSubItem,842150

    .set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,-1)
    With .Items
        With .Add("")
            .Image(EXRADIALMENULib.RadialItemsEnum.exRadialItems) = "favorites.png"
            .Image(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems) =
"download.png"
        End With
    End With
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
```

```
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->PutSubItemsSize(L"48*dpi");
spRadialMenu1->PutPicturesPath(L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
spRadialMenu1-
>PutRadialLineColor(EXRADIALMENULib::exRadialHotItem,RGB(128,128,128));
spRadialMenu1-
>PutRadialLineColor(EXRADIALMENULib::exRadialHotSubItem,RGB(128,128,128));
spRadialMenu1->PutRadialLineColor(EXRADIALMENULib::exRadialHotFullItem,-1);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   EXRADIALMENULib::IItemPtr var_Item = var_Items->Add(L"",vtMissing,vtMissing);
      var_Item->PutImage(EXRADIALMENULib::exRadialItems,"favorites.png");
      var_Item->PutImage(EXRADIALMENULib::exRadialSubItems,"download.png");
```

C++ Builder

```
RadialMenu1->Expanded = true;
RadialMenu1->SubItemsSize = L"48*dpi";
RadialMenu1->PicturesPath = L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
RadialMenu1-
>RadialLineColor[Exradialmenulib_tlb::RadialLineEnum::exRadialHotItem] =
RGB(128,128,128);
RadialMenu1-
>RadialLineColor[Exradialmenulib_tlb::RadialLineEnum::exRadialHotSubItem] =
RGB(128,128,128);
RadialMenu1-
>RadialLineColor[Exradialmenulib_tlb::RadialLineEnum::exRadialHotFullItem] = -1;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   Exradialmenulib_tlb::IItemPtr var_Item = var_Items-
>Add(L"",TNoParam(),TNoParam());
      var_Item-
>set_Image(Exradialmenulib_tlb::RadialItemsEnum::exRadialItems,TVariant("favorites.p

      var_Item-
>set_Image(Exradialmenulib_tlb::RadialItemsEnum::exRadialSubItems,TVariant("downl
```

C#

```
exradialmenu1.Expanded = true;
exradialmenu1.SubItemsSize = "48*dpi";
exradialmenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
exradialmenu1.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRac

exradialmenu1.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRac

exradialmenu1.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exR

exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Item var_Item = var_Items.Add("",null,null);

var_Item.set_Image(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,"favo


var_Item.set_Image(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.Expanded = true;
   RadialMenu1.SubItemsSize = "48*dpi";
   RadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
```

```
   RadialMenu1.RadialLineColor(9) = 8421504;
   RadialMenu1.RadialLineColor(10) = 8421504;
   RadialMenu1.RadialLineColor(11) = -1;
   var var_Items = RadialMenu1.Items;
      var var_Item = var_Items.Add("",null,null);
         var_Item.Image(1) = "favorites.png";
         var_Item.Image(2) = "download.png";
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .Expanded = True
      .SubItemsSize = "48*dpi"
      .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
      .RadialLineColor(9) = RGB(128,128,128)
      .RadialLineColor(10) = RGB(128,128,128)
      .RadialLineColor(11) = -1
      With .Items
         With .Add("")
            .Image(1) = "favorites.png"
            .Image(2) = "download.png"
         End With
      End With
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.Expanded = true;
axRadialMenu1.SubItemsSize = "48*dpi";
axRadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotIt
(uint)ColorTranslator.ToWin32(Color.FromArgb(128,128,128)));
axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotSu
(uint)ColorTranslator.ToWin32(Color.FromArgb(128,128,128)));
axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFu

EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
   EXRADIALMENULib.Item var_Item = var_Items.Add("",null,null);

var_Item.set_Image(EXRADIALMENULib.RadialItemsEnum.exRadialItems,"favorites.png



var_Item.set_Image(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"downloa
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Item,com_Items;
   anytype var_Item,var_Items;
   ;

   super();

   exradialmenu1.Expanded(true);
   exradialmenu1.SubItemsSize("48*dpi");
   exradialmenu1.PicturesPath("C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
```

```
exradialmenu1.RadialLineColor(9/*exRadialHotItem*/,WinApi::RGB2int(128,128,128));

exradialmenu1.RadialLineColor(10/*exRadialHotSubItem*/,WinApi::RGB2int(128,128,12

   exradialmenu1.RadialLineColor(11/*exRadialHotFullItem*/,-1);
   var_Items = exradialmenu1.Items(); com_Items = var_Items;
      var_Item = com_Items.Add(""); com_Item = var_Item;
         com_Item.Image(1/*exRadialItems*/,"favorites.png");
         com_Item.Image(2/*exRadialSubItems*/,"download.png");
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
   Expanded := True;
   SubItemsSize := '48*dpi';
   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotItem,$808080);

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotSubItem,$808080


set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,$ffffffff);

   with Items do
   begin
      with Add('',Nil,Nil) do
      begin
         Image[EXRADIALMENULib.RadialItemsEnum.exRadialItems] := 'favorites.png';
         Image[EXRADIALMENULib.RadialItemsEnum.exRadialSubItems] :=
'download.png';
      end;
   end;
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   Expanded := True;
   SubItemsSize := '48*dpi';
   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
   RadialLineColor[EXRADIALMENULib_TLB.exRadialHotItem] := $808080;
   RadialLineColor[EXRADIALMENULib_TLB.exRadialHotSubItem] := $808080;
   RadialLineColor[EXRADIALMENULib_TLB.exRadialHotFullItem] := $ffffffff;
   with Items do
   begin
      with Add('',Null,Null) do
      begin
         Image[EXRADIALMENULib_TLB.exRadialItems] := 'favorites.png';
         Image[EXRADIALMENULib_TLB.exRadialSubItems] := 'download.png';
      end;
   end;
end
```

VFP

```
with thisform.RadialMenu1
   .Expanded = .T.
   .SubItemsSize = "48*dpi"
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .Object.RadialLineColor(9) = RGB(128,128,128)
   .Object.RadialLineColor(10) = RGB(128,128,128)
   .Object.RadialLineColor(11) = -1
   with .Items
      with .Add("")
         .Image(1) = "favorites.png"
         .Image(2) = "download.png"
      endwith
   endwith
endwith
```

dBASE Plus

```
local oRadialMenu,var_Item,var_Items

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.Expanded = true
oRadialMenu.SubItemsSize = "48*dpi"
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.Template = [RadialLineColor(9) = 8421504] //
oRadialMenu.RadialLineColor(9) = 0x808080
oRadialMenu.Template = [RadialLineColor(10) = 8421504] //
oRadialMenu.RadialLineColor(10) = 0x808080
oRadialMenu.Template = [RadialLineColor(11) = -1] //
oRadialMenu.RadialLineColor(11) = -1
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("")
      // var_Item.Image(1) = "favorites.png"
      with (oRadialMenu)
         TemplateDef = [dim var_Item]
         TemplateDef = var_Item
         Template = [var_Item.Image(1) = "favorites.png"]
      endwith
      // var_Item.Image(2) = "download.png"
      with (oRadialMenu)
         TemplateDef = [dim var_Item]
         TemplateDef = var_Item
         Template = [var_Item.Image(2) = "download.png"]
      endwith
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Item as P
Dim var_Items as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.Expanded = .t.
```

```
oRadialMenu.SubItemsSize = "48*dpi"
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.Template = "RadialLineColor(9) = 8421504" //
oRadialMenu.RadialLineColor(9) = 8421504
oRadialMenu.Template = "RadialLineColor(10) = 8421504" //
oRadialMenu.RadialLineColor(10) = 8421504
oRadialMenu.Template = "RadialLineColor(11) = -1" //
oRadialMenu.RadialLineColor(11) = -1
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("")
      ' var_Item.Image(1) = "favorites.png"
      oRadialMenu.TemplateDef = "dim var_Item"
      oRadialMenu.TemplateDef = var_Item
      oRadialMenu.Template = "var_Item.Image(1) = `favorites.png`"

      ' var_Item.Image(2) = "download.png"
      oRadialMenu.TemplateDef = "dim var_Item"
      oRadialMenu.TemplateDef = var_Item
      oRadialMenu.Template = "var_Item.Image(2) = `download.png`"
```

Visual Objects

```
local var_Item as IItem
local var_Items as IItems

oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:SubItemsSize := "48*dpi"
oDCOCX_Exontrol1:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotItem] := RGB(128,128,128)
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotSubItem] := RGB(128,128,128)
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotFullItem] := -1
var_Items := oDCOCX_Exontrol1:Items
   var_Item := var_Items:Add("",nil,nil)
```

```
    var_Item:[Image,exRadialItems] := "favorites.png"
    var_Item:[Image,exRadialSubItems] := "download.png"
```

PowerBuilder

```
OleObject oRadialMenu,var_Item,var_Items

oRadialMenu = ole_1.Object
oRadialMenu.Expanded = true
oRadialMenu.SubItemsSize = "48*dpi"
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.RadialLineColor(9,RGB(128,128,128))
oRadialMenu.RadialLineColor(10,RGB(128,128,128))
oRadialMenu.RadialLineColor(11,-1)
var_Items = oRadialMenu.Items
  var_Item = var_Items.Add("")
    var_Item.Image(1,"favorites.png")
    var_Item.Image(2,"download.png")
```

Visual DataFlex

```
Procedure OnCreate
  Forward Send OnCreate
  Set ComExpanded to True
  Set ComSubItemsSize to "48*dpi"
  Set ComPicturesPath to "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
  Set ComRadialLineColor OLEexRadialHotItem to (RGB(128,128,128))
  Set ComRadialLineColor OLEexRadialHotSubItem to (RGB(128,128,128))
  Set ComRadialLineColor OLEexRadialHotFullItem to -1
  Variant voItems
  Get ComItems to voItems
  Handle hoItems
  Get Create (RefClass(cComItems)) to hoItems
  Set pvComObject of hoItems to voItems
    Variant voItem
```

```
      Get ComAdd of hoItems "" Nothing Nothing to voItem
      Handle hoItem
      Get Create (RefClass(cComItem)) to hoItem
      Set pvComObject of hoItem to voItem
         Set ComImage of hoItem OLEexRadialItems to "favorites.png"
         Set ComImage of hoItem OLEexRadialSubItems to "download.png"
      Send Destroy to hoItem
   Send Destroy to hoItems
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItem
   LOCAL oItems
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:Expanded := .T.
      oRadialMenu:SubItemsSize := "48*dpi"
      oRadialMenu:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
```

```
oRadialMenu:SetProperty("RadialLineColor",9/*exRadialHotItem*/,AutomationTranslat
 GraMakeRGBColor  ( { 128,128,128 } )  , .F. ))

oRadialMenu:SetProperty("RadialLineColor",10/*exRadialHotSubItem*/,AutomationTra
 GraMakeRGBColor  ( { 128,128,128 } )  , .F. ))
      oRadialMenu:SetProperty("RadialLineColor",11/*exRadialHotFullItem*/,-1)
      oItems := oRadialMenu:Items()
         oItem := oItems:Add("")
            oItem:SetProperty("Image",1/*exRadialItems*/,"favorites.png")
            oItem:SetProperty("Image",2/*exRadialSubItems*/,"download.png")

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

property Item.Index as Long

Retrieves the item's index.

Type	Description
Long	A Long expression that specifies the index of the item in the parent's items collection.

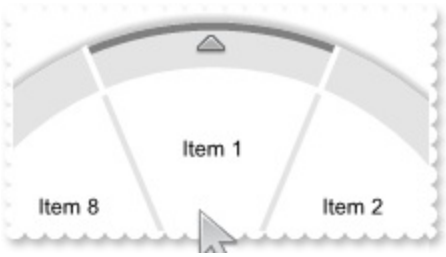
By default, the Index property is defined by the control at adding time. The Index property is read-only. The Add method adds new items to the control. The Items property accesses the child-items collection of the current item. The Parent item property specifies the parent item.

property Item.Items as Items

Retrieves the item's children collection.

Type	Description
Items	An Items collection that specifies the child-items collection.

By default, the item contains no child-items. Use the Items property to access the item's child collection. The Add method adds new child items to the item. By default, the control displays the "arrow" HTML picture, on the sub-items zone, for all items that contains child-items or browse any custom control. The DisplayArrow property indicates whether the "arrow" HTML picture is displayed on the items/sub-items zone of the control, or in both. The DisplayCenterArrow property specifies the ratio to determine where the arrow of items with children is displayed.



The following sample shows how you can add child items:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   With .Items
      With .Add("Item 1").Items
         .Add "SubItem 1"
         .Add "SubItem 2"
      End With
      .Add "Item 2"
      .Add "Item 3"
      .Add "Item 4"
      .Add "Item 5"
      .Add "Item 6"
      .Add "Item 7"
      .Add "Item 8"
   End With
```

```
    .EndUpdate
End With
```

VB6

```
With RadialMenu1
    .BeginUpdate
    .Expanded = True
    With .Items
        With .Add("Item 1").Items
            .Add "SubItem 1"
            .Add "SubItem 2"
        End With
        .Add "Item 2"
        .Add "Item 3"
        .Add "Item 4"
        .Add "Item 5"
        .Add "Item 6"
        .Add "Item 7"
        .Add "Item 8"
    End With
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
    .Expanded = True
    With .Items
        With .Add("Item 1").Items
            .Add("SubItem 1")
            .Add("SubItem 2")
        End With
        .Add("Item 2")
        .Add("Item 3")
        .Add("Item 4")
        .Add("Item 5")
```

```
      .Add("Item 6")
      .Add("Item 7")
      .Add("Item 8")
   End With
   .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .BeginUpdate()
   .Expanded = True
   With .Items
      With .Add("Item 1").Items
         .Add("SubItem 1")
         .Add("SubItem 2")
      End With
      .Add("Item 2")
      .Add("Item 3")
      .Add("Item 4")
      .Add("Item 5")
      .Add("Item 6")
      .Add("Item 7")
      .Add("Item 8")
   End With
   .EndUpdate()
End With
```

C++

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

   #import <ExRadialMenu.dll>
   using namespace EXRADIALMENULib;
*/
```

```
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   EXRADIALMENULib::IItemsPtr var_Items1 = var_Items->Add(L"Item
1",vtMissing,vtMissing)->GetItems();
      var_Items1->Add(L"SubItem 1",vtMissing,vtMissing);
      var_Items1->Add(L"SubItem 2",vtMissing,vtMissing);
   var_Items->Add(L"Item 2",vtMissing,vtMissing);
   var_Items->Add(L"Item 3",vtMissing,vtMissing);
   var_Items->Add(L"Item 4",vtMissing,vtMissing);
   var_Items->Add(L"Item 5",vtMissing,vtMissing);
   var_Items->Add(L"Item 6",vtMissing,vtMissing);
   var_Items->Add(L"Item 7",vtMissing,vtMissing);
   var_Items->Add(L"Item 8",vtMissing,vtMissing);
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   Exradialmenulib_tlb::IItemsPtr var_Items1 = var_Items->Add(L"Item
1",TNoParam(),TNoParam())->Items;
      var_Items1->Add(L"SubItem 1",TNoParam(),TNoParam());
      var_Items1->Add(L"SubItem 2",TNoParam(),TNoParam());
   var_Items->Add(L"Item 2",TNoParam(),TNoParam());
   var_Items->Add(L"Item 3",TNoParam(),TNoParam());
   var_Items->Add(L"Item 4",TNoParam(),TNoParam());
   var_Items->Add(L"Item 5",TNoParam(),TNoParam());
   var_Items->Add(L"Item 6",TNoParam(),TNoParam());
   var_Items->Add(L"Item 7",TNoParam(),TNoParam());
   var_Items->Add(L"Item 8",TNoParam(),TNoParam());
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Items var_Items1 = var_Items.Add("Item
1",null,null).Items;
      var_Items1.Add("SubItem 1",null,null);
      var_Items1.Add("SubItem 2",null,null);
   var_Items.Add("Item 2",null,null);
   var_Items.Add("Item 3",null,null);
   var_Items.Add("Item 4",null,null);
   var_Items.Add("Item 5",null,null);
   var_Items.Add("Item 6",null,null);
   var_Items.Add("Item 7",null,null);
   var_Items.Add("Item 8",null,null);
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.Expanded = true;
   var var_Items = RadialMenu1.Items;
      var var_Items1 = var_Items.Add("Item 1",null,null).Items;
         var_Items1.Add("SubItem 1",null,null);
         var_Items1.Add("SubItem 2",null,null);
      var_Items.Add("Item 2",null,null);
      var_Items.Add("Item 3",null,null);
      var_Items.Add("Item 4",null,null);
      var_Items.Add("Item 5",null,null);
```

```
      var_Items.Add("Item 6",null,null);
      var_Items.Add("Item 7",null,null);
      var_Items.Add("Item 8",null,null);
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .Expanded = True
      With .Items
         With .Add("Item 1").Items
            .Add "SubItem 1"
            .Add "SubItem 2"
         End With
         .Add "Item 2"
         .Add "Item 3"
         .Add "Item 4"
         .Add "Item 5"
         .Add "Item 6"
         .Add "Item 7"
         .Add "Item 8"
      End With
      .EndUpdate
   End With
End Function
</SCRIPT>
```

```
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
   EXRADIALMENULib.Items var_Items1 = var_Items.Add("Item 1",null,null).Items;
      var_Items1.Add("SubItem 1",null,null);
      var_Items1.Add("SubItem 2",null,null);
   var_Items.Add("Item 2",null,null);
   var_Items.Add("Item 3",null,null);
   var_Items.Add("Item 4",null,null);
   var_Items.Add("Item 5",null,null);
   var_Items.Add("Item 6",null,null);
   var_Items.Add("Item 7",null,null);
   var_Items.Add("Item 8",null,null);
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Item,com_Items,com_Items1;
   anytype var_Item,var_Items,var_Items1;
   ;

   super();

   exradialmenu1.BeginUpdate();
   exradialmenu1.Expanded(true);
   var_Items = exradialmenu1.Items(); com_Items = var_Items;
      var_Item = COM::createFromObject(com_Items.Add("Item 1")); com_Item =
var_Item;
      var_Items1 = com_Item.Items(); com_Items1 = var_Items1;
         com_Items1.Add("SubItem 1");
```

```
        com_Items1.Add("SubItem 2");
    com_Items.Add("Item 2");
    com_Items.Add("Item 3");
    com_Items.Add("Item 4");
    com_Items.Add("Item 5");
    com_Items.Add("Item 6");
    com_Items.Add("Item 7");
    com_Items.Add("Item 8");
  exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
  BeginUpdate();
  Expanded := True;
  with Items do
  begin
    with Add('Item 1',Nil,Nil).Items do
    begin
      Add('SubItem 1',Nil,Nil);
      Add('SubItem 2',Nil,Nil);
    end;
    Add('Item 2',Nil,Nil);
    Add('Item 3',Nil,Nil);
    Add('Item 4',Nil,Nil);
    Add('Item 5',Nil,Nil);
    Add('Item 6',Nil,Nil);
    Add('Item 7',Nil,Nil);
    Add('Item 8',Nil,Nil);
  end;
  EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
```

```
begin
   BeginUpdate();
   Expanded := True;
   with Items do
   begin
      with Add('Item 1',Null,Null).Items do
      begin
         Add('SubItem 1',Null,Null);
         Add('SubItem 2',Null,Null);
      end;
      Add('Item 2',Null,Null);
      Add('Item 3',Null,Null);
      Add('Item 4',Null,Null);
      Add('Item 5',Null,Null);
      Add('Item 6',Null,Null);
      Add('Item 7',Null,Null);
      Add('Item 8',Null,Null);
   end;
   EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
   .BeginUpdate
   .Expanded = .T.
   with .Items
      with .Add("Item 1").Items
         .Add("SubItem 1")
         .Add("SubItem 2")
      endwith
      .Add("Item 2")
      .Add("Item 3")
      .Add("Item 4")
      .Add("Item 5")
      .Add("Item 6")
      .Add("Item 7")
```

```
      .Add("Item 8")
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu,var_Items,var_Items1

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Items as P
Dim var_Items1 as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = .t.
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
```

```
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

Visual Objects

```
local var_Items,var_Items1 as IItems

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
var_Items := oDCOCX_Exontrol1:Items
   var_Items1 := var_Items:Add("Item 1",nil,nil):Items
      var_Items1:Add("SubItem 1",nil,nil)
      var_Items1:Add("SubItem 2",nil,nil)
   var_Items:Add("Item 2",nil,nil)
   var_Items:Add("Item 3",nil,nil)
   var_Items:Add("Item 4",nil,nil)
   var_Items:Add("Item 5",nil,nil)
   var_Items:Add("Item 6",nil,nil)
   var_Items:Add("Item 7",nil,nil)
   var_Items:Add("Item 8",nil,nil)
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu,var_Items,var_Items1

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
```

```
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComExpanded to True
   Variant voItems
   Get ComItems to voItems
   Handle hoItems
   Get Create (RefClass(cComItems)) to hoItems
   Set pvComObject of hoItems to voItems
      Variant voItem
      Get ComAdd of hoItems "Item 1" Nothing Nothing to voItem
      Handle hoItem
      Get Create (RefClass(cComItem)) to hoItem
      Set pvComObject of hoItem to voItem
         Variant voItems1
         Get ComItems of hoItem to voItems1
         Handle hoItems1
         Get Create (RefClass(cComItems)) to hoItems1
         Set pvComObject of hoItems1 to voItems1
            Get ComAdd of hoItems1 "SubItem 1" Nothing Nothing to Nothing
            Get ComAdd of hoItems1 "SubItem 2" Nothing Nothing to Nothing
```

```
        Send Destroy to hoItems1
    Send Destroy to hoItem
    Get ComAdd of hoItems "Item 2" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 3" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 4" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 5" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 6" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 7" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 8" Nothing Nothing to Nothing
  Send Destroy to hoItems
  Send ComEndUpdate
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItems,oItems1
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
      oItems := oRadialMenu:Items()
```

```
      oItems1 := oItems:Add("Item 1"):Items()
         oItems1:Add("SubItem 1")
         oItems1:Add("SubItem 2")
      oItems:Add("Item 2")
      oItems:Add("Item 3")
      oItems:Add("Item 4")
      oItems:Add("Item 5")
      oItems:Add("Item 6")
      oItems:Add("Item 7")
      oItems:Add("Item 8")
   oRadialMenu:EndUpdate()

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

property Item.Name as String

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

Type	Description
String	A String expression that specifies the caption of the item, for items zone. The Name property supports HTML format as described bellow.

By default, the Name property is empty. The Name property retrieves or sets a value that indicates the item's name. The Name property of the Item object is equivalent with the Caption(exRadialItems) property. The Caption property retrieves or sets a value that indicates the item's caption. You can specify the caption of the item using the Caption parameter of the Add method. The ForeColor property specifies the item's foreground color. The Image property assigns an icon/picture to the item.

The Name property supports the following built-in HTML format:

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

color in hexa values.

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

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

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

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

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

property Item.Parent as Item

Retrieves the parent of the item.

Type	Description
Item	An Item object that specifies the parent's item.

The Parent item property specifies the parent item. The Add method adds new items (child) to the control. The Items property accesses the child-items collection of the current item. The Root property of the control accesses the root item. The root item has no parent item. The SelectParent event occurs once the user clicks the parent of the item. The BrowseItem event notifies when a new item has been selected / browsed.

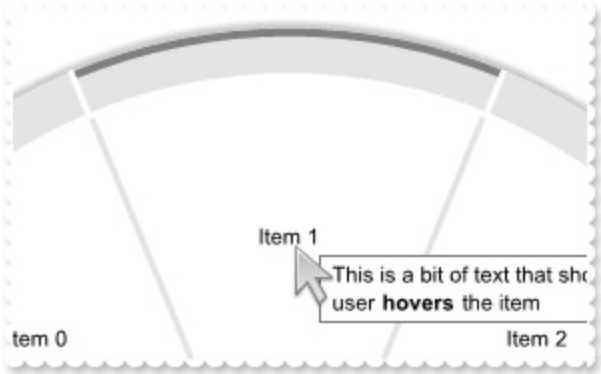
property Item.Tooltip(Type as RadialItemsEnum) as String

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

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies the part of the item to assign the tooltip.
String	A String expression that defines the item's tooltip. The ToolTip supports built-in HTML format.

By default, the Tooltip property is empty. The Tooltip property specifies the item's tooltip. The TooltipTitle property retrieves or sets a value that indicates the title of the item's tooltip. Use the ShowToolTip method to display a custom tooltip. The ItemFromPoint property returns the item from the cursor. Use the ToolTipWidth property to specify the width of the tooltip window Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

The following screen shot shows a tooltip when user hovers the mouse over an item:



The ToolTip supports the following built-in HTML format:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning

and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

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

(the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold

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

gradient-center

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

outlined

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

shadow

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

The following samples show how you can assign a tooltip to an item:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .DisplayAngle = -45
   .Expanded = True
   With .Items
      .ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
      .Add("Item 1").Tooltip(1) = "This is a bit of text that shown when user
<b>hovers</b> the item"
      With .Add("Item 2")
         .Tooltip(1) = "This is a bit of text that shown when user hovers the item"
         .Tooltip(2) = "This is a bit of text that shown when user hovers the sub-item"
      End With
   End With
End With
```

VB6

```
With RadialMenu1
   .DisplayAngle = -45
   .Expanded = True
   With .Items
      .ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
      .Add("Item 1").Tooltip(exRadialItems) = "This is a bit of text that shown when user
<b>hovers</b> the item"
      With .Add("Item 2")
         .Tooltip(exRadialItems) = "This is a bit of text that shown when user hovers the
item"
         .Tooltip(exRadialSubItems) = "This is a bit of text that shown when user hovers
the sub-item"
      End With
   End With
End With
```

VB.NET

```
With Exradialmenu1
   .DisplayAngle = -45
   .Expanded = True
   With .Items
      .ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
      .Add("Item
1").set_Tooltip(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,"This is a
bit of text that shown when user <b>hovers</b> the item")
      With .Add("Item 2")
         .set_Tooltip(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,"This
is a bit of text that shown when user hovers the item")

.set_Tooltip(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"This is a
bit of text that shown when user hovers the sub-item")
      End With
   End With
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .DisplayAngle = -45
   .Expanded = True
   With .Items
      .ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
      .Add("Item 1").Tooltip(EXRADIALMENULib.RadialItemsEnum.exRadialItems) =
"This is a bit of text that shown when user <b>hovers</b> the item"
      With .Add("Item 2")
         .Tooltip(EXRADIALMENULib.RadialItemsEnum.exRadialItems) = "This is a bit of
text that shown when user hovers the item"
         .Tooltip(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems) = "This is a
bit of text that shown when user hovers the sub-item"
      End With
   End With
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->PutDisplayAngle(-45);
spRadialMenu1->PutExpanded(VARIANT_TRUE);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   var_Items->PutToString(L"Item 0[ttp=tooltip's item][sttp=tooltip's subitem]");
   var_Items->Add(L"Item 1",vtMissing,vtMissing)-
>PutTooltip(EXRADIALMENULib::exRadialItems,L"This is a bit of text that shown when
user <b>hovers</b> the item");
   EXRADIALMENULib::IItemPtr var_Item = var_Items->Add(L"Item
2",vtMissing,vtMissing);
      var_Item->PutTooltip(EXRADIALMENULib::exRadialItems,L"This is a bit of text
that shown when user hovers the item");
      var_Item->PutTooltip(EXRADIALMENULib::exRadialSubItems,L"This is a bit of text
that shown when user hovers the sub-item");
```

C++ Builder

```
RadialMenu1->DisplayAngle = -45;
RadialMenu1->Expanded = true;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   var_Items->ToString = L"Item 0[ttp=tooltip's item][sttp=tooltip's subitem]";
   var_Items->Add(L"Item 1",TNoParam(),TNoParam())-
>set_Tooltip(Exradialmenulib_tlb::RadialItemsEnum::exRadialItems,L"This is a bit of
text that shown when user <b>hovers</b> the item");
   Exradialmenulib_tlb::IItemPtr var_Item = var_Items->Add(L"Item
2",TNoParam(),TNoParam());
```

```
	var_Item->set_Tooltip(ExradialmenuLib_tlb::RadialItemsEnum::exRadialItems,L"This is a bit of text that shown when user hovers the item");
	var_Item->set_Tooltip(ExradialmenuLib_tlb::RadialItemsEnum::exRadialSubItems,L"This is a bit of text that shown when user hovers the sub-item");
```

C#

```
exradialmenu1.DisplayAngle = -45;
exradialmenu1.Expanded = true;
exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   var_Items.ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]";
   var_Items.Add("Item 1",null,null).set_Tooltip(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,"Th
 is a bit of text that shown when user <b>hovers</b> the item");
   exontrol.EXRADIALMENULib.Item var_Item = var_Items.Add("Item 2",null,null);

var_Item.set_Tooltip(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,"This
 is a bit of text that shown when user hovers the item");

var_Item.set_Tooltip(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"
 is a bit of text that shown when user hovers the sub-item");
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.DisplayAngle = -45;
   RadialMenu1.Expanded = true;
   var var_Items = RadialMenu1.Items;
```

```
    var_Items.ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]";
    var_Items.Add("Item 1",null,null).Tooltip(1) = "This is a bit of text that shown when user <b>hovers</b> the item";
    var var_Item = var_Items.Add("Item 2",null,null);
      var_Item.Tooltip(1) = "This is a bit of text that shown when user hovers the item";
      var_Item.Tooltip(2) = "This is a bit of text that shown when user hovers the sub-item";
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C" id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
  With RadialMenu1
    .DisplayAngle = -45
    .Expanded = True
    With .Items
      .ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
      .Add("Item 1").Tooltip(1) = "This is a bit of text that shown when user <b>hovers</b> the item"
      With .Add("Item 2")
        .Tooltip(1) = "This is a bit of text that shown when user hovers the item"
        .Tooltip(2) = "This is a bit of text that shown when user hovers the sub-item"
      End With
    End With
  End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.DisplayAngle = -45;
axRadialMenu1.Expanded = true;
EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
   var_Items.ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]";
   var_Items.Add("Item
1",null,null).set_Tooltip(EXRADIALMENULib.RadialItemsEnum.exRadialItems,"This is a
bit of text that shown when user <b>hovers</b> the item");
   EXRADIALMENULib.Item var_Item = var_Items.Add("Item 2",null,null);
      var_Item.set_Tooltip(EXRADIALMENULib.RadialItemsEnum.exRadialItems,"This is
a bit of text that shown when user hovers the item");

var_Item.set_Tooltip(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,"This is a
bit of text that shown when user hovers the sub-item");
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Item,com_Items;
   anytype var_Item,var_Items;
   ;

   super();

   exradialmenu1.DisplayAngle(-45);
   exradialmenu1.Expanded(true);
   var_Items = exradialmenu1.Items(); com_Items = var_Items;
      com_Items.ToString("Item 0[ttp=tooltip's item][sttp=tooltip's subitem]");
      var_Item = COM::createFromObject(com_Items.Add("Item 1")); com_Item =
var_Item;
      com_Item.Tooltip(1/*exRadialItems*/,"This is a bit of text that shown when user
<b>hovers</b> the item");
      var_Item = com_Items.Add("Item 2"); com_Item = var_Item;
```

```
        com_Item.Tooltip(1/*exRadialItems*/,"This is a bit of text that shown when user hovers the item");
        com_Item.Tooltip(2/*exRadialSubItems*/,"This is a bit of text that shown when user hovers the sub-item");
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
  DisplayAngle := -45;
  Expanded := True;
  with Items do
  begin
    ToString := 'Item 0[ttp=tooltip''s item][sttp=tooltip''s subitem]';
    Add('Item 1',Nil,Nil).Tooltip[EXRADIALMENULib.RadialItemsEnum.exRadialItems] := 'This is a bit of text that shown when user <b>hovers</b> the item';
    with Add('Item 2',Nil,Nil) do
    begin
      Tooltip[EXRADIALMENULib.RadialItemsEnum.exRadialItems] := 'This is a bit of text that shown when user hovers the item';
      Tooltip[EXRADIALMENULib.RadialItemsEnum.exRadialSubItems] := 'This is a bit of text that shown when user hovers the sub-item';
    end;
  end;
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
  DisplayAngle := -45;
  Expanded := True;
  with Items do
  begin
    ToString := 'Item 0[ttp=tooltip''s item][sttp=tooltip''s subitem]';
    Add('Item 1',Null,Null).Tooltip[EXRADIALMENULib_TLB.exRadialItems] := 'This is a bit of text that shown when user <b>hovers</b> the item';
```

```
      with Add('Item 2',Null,Null) do
      begin
        Tooltip[EXRADIALMENULib_TLB.exRadialItems] := 'This is a bit of text that
shown when user hovers the item';
        Tooltip[EXRADIALMENULib_TLB.exRadialSubItems] := 'This is a bit of text that
shown when user hovers the sub-item';
      end;
  end;
end
```

VFP

```
with thisform.RadialMenu1
  .DisplayAngle = -45
  .Expanded = .T.
  with .Items
    .ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
    .Add("Item 1").Tooltip(1) = "This is a bit of text that shown when user
<b>hovers</b> the item"
    with .Add("Item 2")
      .Tooltip(1) = "This is a bit of text that shown when user hovers the item"
      .Tooltip(2) = "This is a bit of text that shown when user hovers the sub-item"
    endwith
  endwith
endwith
```

dBASE Plus

```
local oRadialMenu,var_Item,var_Item1,var_Items

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.DisplayAngle = -45
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
  var_Items.ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
  // var_Items.Add("Item 1").Tooltip(1) = "This is a bit of text that shown when user
<b>hovers</b> the item"
  var_Item = var_Items.Add("Item 1")
```

```
  with (oRadialMenu)
    TemplateDef = [dim var_Item]
    TemplateDef = var_Item
    Template = [var_Item.Tooltip(1) = "This is a bit of text that shown when user
<b>hovers</b> the item"]
  endwith
  var_Item1 = var_Items.Add("Item 2")
    // var_Item1.Tooltip(1) = "This is a bit of text that shown when user hovers the
item"
    with (oRadialMenu)
      TemplateDef = [dim var_Item1]
      TemplateDef = var_Item1
      Template = [var_Item1.Tooltip(1) = "This is a bit of text that shown when user
hovers the item"]
    endwith
    // var_Item1.Tooltip(2) = "This is a bit of text that shown when user hovers the
sub-item"
    with (oRadialMenu)
      TemplateDef = [dim var_Item1]
      TemplateDef = var_Item1
      Template = [var_Item1.Tooltip(2) = "This is a bit of text that shown when user
hovers the sub-item"]
    endwith
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Item as local
Dim var_Item1 as P
Dim var_Items as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.DisplayAngle = -45
oRadialMenu.Expanded = .t.
var_Items = oRadialMenu.Items
  var_Items.ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
```

```
    ' var_Items.Add("Item 1").Tooltip(1) = "This is a bit of text that shown when user <b>hovers</b> the item"
    var_Item = var_Items.Add("Item 1")
    oRadialMenu.TemplateDef = "dim var_Item"
    oRadialMenu.TemplateDef = var_Item
    oRadialMenu.Template = "var_Item.Tooltip(1) = `This is a bit of text that shown when user <b>hovers</b> the item`"

    var_Item1 = var_Items.Add("Item 2")
        ' var_Item1.Tooltip(1) = "This is a bit of text that shown when user hovers the item"
        oRadialMenu.TemplateDef = "dim var_Item1"
        oRadialMenu.TemplateDef = var_Item1
        oRadialMenu.Template = "var_Item1.Tooltip(1) = `This is a bit of text that shown when user hovers the item`"

        ' var_Item1.Tooltip(2) = "This is a bit of text that shown when user hovers the sub-item"
        oRadialMenu.TemplateDef = "dim var_Item1"
        oRadialMenu.TemplateDef = var_Item1
        oRadialMenu.Template = "var_Item1.Tooltip(2) = `This is a bit of text that shown when user hovers the sub-item`"
```

Visual Objects

```
local var_Item as IItem
local var_Items as IItems

oDCOCX_Exontrol1:DisplayAngle := -45
oDCOCX_Exontrol1:Expanded := true
var_Items := oDCOCX_Exontrol1:Items
    var_Items:ToString := "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
    var_Items:Add("Item 1",nil,nil):[Tooltip,exRadialItems] := "This is a bit of text that shown when user <b>hovers</b> the item"
    var_Item := var_Items:Add("Item 2",nil,nil)
```

```
    var_Item:[Tooltip,exRadialItems] := "This is a bit of text that shown when user hovers the item"
    var_Item:[Tooltip,exRadialSubItems] := "This is a bit of text that shown when user hovers the sub-item"
```

PowerBuilder

```
OleObject oRadialMenu,var_Item,var_Items

oRadialMenu = ole_1.Object
oRadialMenu.DisplayAngle = -45
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
  var_Items.ToString = "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
  var_Items.Add("Item 1").Tooltip(1,"This is a bit of text that shown when user <b>hovers</b> the item")
  var_Item = var_Items.Add("Item 2")
    var_Item.Tooltip(1,"This is a bit of text that shown when user hovers the item")
    var_Item.Tooltip(2,"This is a bit of text that shown when user hovers the sub-item")
```

Visual DataFlex

```
Procedure OnCreate
  Forward Send OnCreate
  Set ComDisplayAngle to -45
  Set ComExpanded to True
  Variant voItems
  Get ComItems to voItems
  Handle hoItems
  Get Create (RefClass(cComItems)) to hoItems
  Set pvComObject of hoItems to voItems
    Set ComToString of hoItems to "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
    Variant voItem
    Get ComAdd of hoItems "Item 1" Nothing Nothing to voItem
```

```
        Handle hoItem
        Get Create (RefClass(cComItem)) to hoItem
        Set pvComObject of hoItem to voItem
            Set ComTooltip of hoItem OLEexRadialItems to "This is a bit of text that shown
when user <b>hovers</b> the item"
        Send Destroy to hoItem
        Variant voItem1
        Get ComAdd of hoItems "Item 2" Nothing Nothing to voItem1
        Handle hoItem1
        Get Create (RefClass(cComItem)) to hoItem1
        Set pvComObject of hoItem1 to voItem1
            Set ComTooltip of hoItem1 OLEexRadialItems to "This is a bit of text that
shown when user hovers the item"
            Set ComTooltip of hoItem1 OLEexRadialSubItems to "This is a bit of text that
shown when user hovers the sub-item"
        Send Destroy to hoItem1
    Send Destroy to hoItems
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItem
   LOCAL oItems
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
```

```
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:DisplayAngle := -45
      oRadialMenu:Expanded := .T.
      oItems := oRadialMenu:Items()
         oItems:ToString := "Item 0[ttp=tooltip's item][sttp=tooltip's subitem]"
         oItems:Add("Item 1"):SetProperty("Tooltip",1/*exRadialItems*/,"This is a bit of text that shown when user <b>hovers</b> the item")
         oItem := oItems:Add("Item 2")
            oItem:SetProperty("Tooltip",1/*exRadialItems*/,"This is a bit of text that shown when user hovers the item")
            oItem:SetProperty("Tooltip",2/*exRadialSubItems*/,"This is a bit of text that shown when user hovers the sub-item")

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

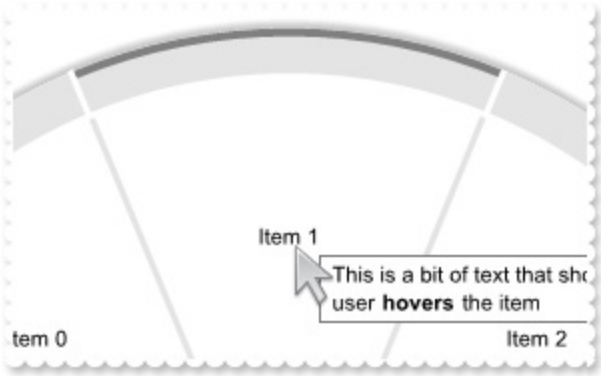
property Item.TooltipTitle(Type as RadialItemsEnum) as String

Retrieves or sets a value that indicates the title of the item's tooltip.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies the part of the item to assign the title.
String	A String expression that defines the title of the item's tooltip.

By default, the TooltipTitle property is empty. The TooltipTitle property retrieves or sets a value that indicates the title of the item's tooltip. The Tooltip property specifies the item's tooltip. Use the ShowToolTip method to display a custom tooltip. The ItemFromPoint property returns the item from the cursor. Use the ToolTipWidth property to specify the width of the tooltip window Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

The following screen shot shows a tooltip when user hovers the mouse over an item:



property Item.UserData(Type as RadialItemsEnum) as Variant

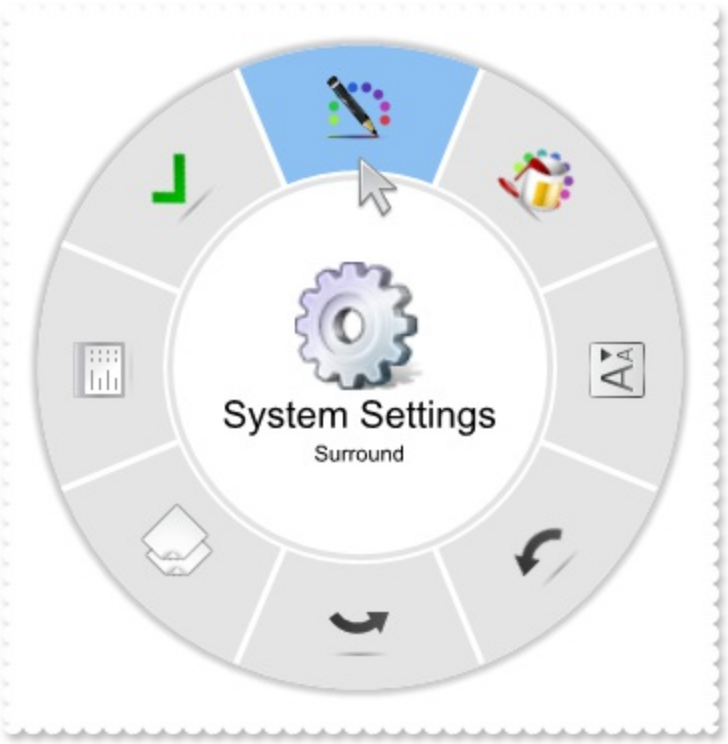
Retrieves or sets a value that indicates the item's user data.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies the part of the item to be changed.
Variant	A VARIANT expression that specifies any extra data associated with the item.

By default, the UserData property is empty. The UserData property retrieves or sets a value that indicates the item's user data. The Caption property retrieves or sets a value that indicates the item's caption.

Items object

The Items collection holds the radial menu's items collection. The Items collection holds Item objects. The Items property of the control, accesses the control's Items collection.



The Items object supports the following properties and methods:

Name	Description
Add	Adds an Item object and returns a reference to the newly created object.
Clear	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
Item	Returns a specific Item object giving its index or name.
Remove	Removes a specific member from the collection.
ToString	Loads or saves the Items collection using string representation.

method Items.Add (Caption as String, [Image as Variant], [Type 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 item's caption to be added. The Caption parameter supports HTML as described in the Caption property.
Image as Variant	A VARIANT expression that specifies the image to be shown on the item. The Image parameter could be: • A String expression indicates: ◦ a name of a picture file in the PicturePath folder. For instance, Image(exRadialItems) = "favorites.png", loads the favorites.png file if found in the PicturePath folder. ◦ a picture file including its absolute path. For instance, Image(exRadialItems) = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor loads the favorites.png file from absolute path ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, Image(exRadialItems) = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on. • A Picture object that indicates the picture to be displayed. For instance, Image(exRadialItems) = LoadPicture("picture.jpg") • a long/string expression that specifies the index of the icon to be displayed (0-based). The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. If no icon/picture/image is found, the item displays no icon/picture/image.

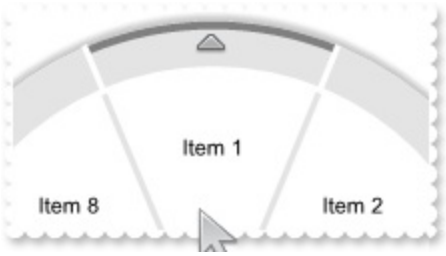
Type as Variant	A RadialItemsEnum expression that specifies the portion of the item to be changed. If missing, the exRadialItems value is used instead, so the items portion of the item is changed.

Return	Description
Item	An Item object being added.

The user can add new items to the control using any of the following:

- Add method, adds a new item to the control. The Add method can be used to add child-items as well.
- ToString property of the Items collection, loads or saves the Items collection using string representation.
- ToString property of the control, loads or saves the Items collection using string representation.

The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control.



The Clear method clears all items from the collection. The Count property specifies the number of items in the collection. The Item property accesses an item based on its index or name. The Remove method removes an item from the collection.

The following samples show how you can add new items / child items to the control:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   With .Items
      With .Add("Item 1").Items
         .Add "SubItem 1"
```

```
            .Add "SubItem 2"
        End With
        .Add "Item 2"
        .Add "Item 3"
        .Add "Item 4"
        .Add "Item 5"
        .Add "Item 6"
        .Add "Item 7"
        .Add "Item 8"
    End With
    .EndUpdate
End With
```

VB6

```
With RadialMenu1
    .BeginUpdate
    .Expanded = True
    With .Items
        With .Add("Item 1").Items
            .Add "SubItem 1"
            .Add "SubItem 2"
        End With
        .Add "Item 2"
        .Add "Item 3"
        .Add "Item 4"
        .Add "Item 5"
        .Add "Item 6"
        .Add "Item 7"
        .Add "Item 8"
    End With
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
```

```
    .Expanded = True
    With .Items
        With .Add("Item 1").Items
            .Add("SubItem 1")
            .Add("SubItem 2")
        End With
        .Add("Item 2")
        .Add("Item 3")
        .Add("Item 4")
        .Add("Item 5")
        .Add("Item 6")
        .Add("Item 7")
        .Add("Item 8")
    End With
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
    .BeginUpdate()
    .Expanded = True
    With .Items
        With .Add("Item 1").Items
            .Add("SubItem 1")
            .Add("SubItem 2")
        End With
        .Add("Item 2")
        .Add("Item 3")
        .Add("Item 4")
        .Add("Item 5")
        .Add("Item 6")
        .Add("Item 7")
        .Add("Item 8")
    End With
    .EndUpdate()
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   EXRADIALMENULib::IItemsPtr var_Items1 = var_Items->Add(L"Item
1",vtMissing,vtMissing)->GetItems();
      var_Items1->Add(L"SubItem 1",vtMissing,vtMissing);
      var_Items1->Add(L"SubItem 2",vtMissing,vtMissing);
   var_Items->Add(L"Item 2",vtMissing,vtMissing);
   var_Items->Add(L"Item 3",vtMissing,vtMissing);
   var_Items->Add(L"Item 4",vtMissing,vtMissing);
   var_Items->Add(L"Item 5",vtMissing,vtMissing);
   var_Items->Add(L"Item 6",vtMissing,vtMissing);
   var_Items->Add(L"Item 7",vtMissing,vtMissing);
   var_Items->Add(L"Item 8",vtMissing,vtMissing);
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   Exradialmenulib_tlb::IItemsPtr var_Items1 = var_Items->Add(L"Item
1",TNoParam(),TNoParam())->Items;
      var_Items1->Add(L"SubItem 1",TNoParam(),TNoParam());
      var_Items1->Add(L"SubItem 2",TNoParam(),TNoParam());
```

```
   var_Items->Add(L"Item 2",TNoParam(),TNoParam());
   var_Items->Add(L"Item 3",TNoParam(),TNoParam());
   var_Items->Add(L"Item 4",TNoParam(),TNoParam());
   var_Items->Add(L"Item 5",TNoParam(),TNoParam());
   var_Items->Add(L"Item 6",TNoParam(),TNoParam());
   var_Items->Add(L"Item 7",TNoParam(),TNoParam());
   var_Items->Add(L"Item 8",TNoParam(),TNoParam());
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Items var_Items1 = var_Items.Add("Item
1",null,null).Items;
      var_Items1.Add("SubItem 1",null,null);
      var_Items1.Add("SubItem 2",null,null);
   var_Items.Add("Item 2",null,null);
   var_Items.Add("Item 3",null,null);
   var_Items.Add("Item 4",null,null);
   var_Items.Add("Item 5",null,null);
   var_Items.Add("Item 6",null,null);
   var_Items.Add("Item 7",null,null);
   var_Items.Add("Item 8",null,null);
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
```

```
   RadialMenu1.BeginUpdate();
   RadialMenu1.Expanded = true;
   var var_Items = RadialMenu1.Items;
      var var_Items1 = var_Items.Add("Item 1",null,null).Items;
         var_Items1.Add("SubItem 1",null,null);
         var_Items1.Add("SubItem 2",null,null);
      var_Items.Add("Item 2",null,null);
      var_Items.Add("Item 3",null,null);
      var_Items.Add("Item 4",null,null);
      var_Items.Add("Item 5",null,null);
      var_Items.Add("Item 6",null,null);
      var_Items.Add("Item 7",null,null);
      var_Items.Add("Item 8",null,null);
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .Expanded = True
      With .Items
         With .Add("Item 1").Items
            .Add "SubItem 1"
            .Add "SubItem 2"
         End With
         .Add "Item 2"
         .Add "Item 3"
```

```
        .Add "Item 4"
        .Add "Item 5"
        .Add "Item 6"
        .Add "Item 7"
        .Add "Item 8"
    End With
    .EndUpdate
  End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
  EXRADIALMENULib.Items var_Items1 = var_Items.Add("Item 1",null,null).Items;
    var_Items1.Add("SubItem 1",null,null);
    var_Items1.Add("SubItem 2",null,null);
  var_Items.Add("Item 2",null,null);
  var_Items.Add("Item 3",null,null);
  var_Items.Add("Item 4",null,null);
  var_Items.Add("Item 5",null,null);
  var_Items.Add("Item 6",null,null);
  var_Items.Add("Item 7",null,null);
  var_Items.Add("Item 8",null,null);
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
  COM com_Item,com_Items,com_Items1;
  anytype var_Item,var_Items,var_Items1;
  ;
```

```
    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.Expanded(true);
    var_Items = exradialmenu1.Items(); com_Items = var_Items;
        var_Item = COM::createFromObject(com_Items.Add("Item 1")); com_Item =
var_Item;
        var_Items1 = com_Item.Items(); com_Items1 = var_Items1;
            com_Items1.Add("SubItem 1");
            com_Items1.Add("SubItem 2");
        com_Items.Add("Item 2");
        com_Items.Add("Item 3");
        com_Items.Add("Item 4");
        com_Items.Add("Item 5");
        com_Items.Add("Item 6");
        com_Items.Add("Item 7");
        com_Items.Add("Item 8");
    exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
    BeginUpdate();
    Expanded := True;
    with Items do
    begin
        with Add('Item 1',Nil,Nil).Items do
        begin
            Add('SubItem 1',Nil,Nil);
            Add('SubItem 2',Nil,Nil);
        end;
        Add('Item 2',Nil,Nil);
        Add('Item 3',Nil,Nil);
        Add('Item 4',Nil,Nil);
```

```
      Add('Item 5',Nil,Nil);
      Add('Item 6',Nil,Nil);
      Add('Item 7',Nil,Nil);
      Add('Item 8',Nil,Nil);
    end;
    EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
    BeginUpdate();
    Expanded := True;
    with Items do
    begin
      with Add('Item 1',Null,Null).Items do
      begin
        Add('SubItem 1',Null,Null);
        Add('SubItem 2',Null,Null);
      end;
      Add('Item 2',Null,Null);
      Add('Item 3',Null,Null);
      Add('Item 4',Null,Null);
      Add('Item 5',Null,Null);
      Add('Item 6',Null,Null);
      Add('Item 7',Null,Null);
      Add('Item 8',Null,Null);
    end;
    EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
    .BeginUpdate
    .Expanded = .T.
    with .Items
```

```
      with .Add("Item 1").Items
         .Add("SubItem 1")
         .Add("SubItem 2")
      endwith
      .Add("Item 2")
      .Add("Item 3")
      .Add("Item 4")
      .Add("Item 5")
      .Add("Item 6")
      .Add("Item 7")
      .Add("Item 8")
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu,var_Items,var_Items1

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Items as P
Dim var_Items1 as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = .t.
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

Visual Objects

```
local var_Items,var_Items1 as lItems

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
var_Items := oDCOCX_Exontrol1:Items
   var_Items1 := var_Items:Add("Item 1",nil,nil):Items
      var_Items1:Add("SubItem 1",nil,nil)
      var_Items1:Add("SubItem 2",nil,nil)
   var_Items:Add("Item 2",nil,nil)
   var_Items:Add("Item 3",nil,nil)
   var_Items:Add("Item 4",nil,nil)
   var_Items:Add("Item 5",nil,nil)
   var_Items:Add("Item 6",nil,nil)
   var_Items:Add("Item 7",nil,nil)
```

```
   var_Items:Add("Item 8",nil,nil)
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu,var_Items,var_Items1

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
var_Items = oRadialMenu.Items
   var_Items1 = var_Items.Add("Item 1").Items
      var_Items1.Add("SubItem 1")
      var_Items1.Add("SubItem 2")
   var_Items.Add("Item 2")
   var_Items.Add("Item 3")
   var_Items.Add("Item 4")
   var_Items.Add("Item 5")
   var_Items.Add("Item 6")
   var_Items.Add("Item 7")
   var_Items.Add("Item 8")
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComExpanded to True
   Variant voItems
   Get ComItems to voItems
   Handle hoItems
   Get Create (RefClass(cComItems)) to hoItems
   Set pvComObject of hoItems to voItems
      Variant voItem
      Get ComAdd of hoItems "Item 1" Nothing Nothing to voItem
```

```
    Handle hoItem
    Get Create (RefClass(cComItem)) to hoItem
    Set pvComObject of hoItem to voItem
      Variant voItems1
      Get ComItems of hoItem to voItems1
      Handle hoItems1
      Get Create (RefClass(cComItems)) to hoItems1
      Set pvComObject of hoItems1 to voItems1
        Get ComAdd of hoItems1 "SubItem 1" Nothing Nothing to Nothing
        Get ComAdd of hoItems1 "SubItem 2" Nothing Nothing to Nothing
      Send Destroy to hoItems1
    Send Destroy to hoItem
    Get ComAdd of hoItems "Item 2" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 3" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 4" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 5" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 6" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 7" Nothing Nothing to Nothing
    Get ComAdd of hoItems "Item 8" Nothing Nothing to Nothing
  Send Destroy to hoItems
  Send ComEndUpdate
End_Procedure
```

XBase++

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

PROCEDURE Main
  LOCAL oForm
  LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
  LOCAL oItems,oItems1
  LOCAL oRadialMenu

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

```
   oForm:close  := {|| PostAppEvent( xbeP_Quit )}

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
      oItems := oRadialMenu:Items()
         oItems1 := oItems:Add("Item 1"):Items()
            oItems1:Add("SubItem 1")
            oItems1:Add("SubItem 2")
         oItems:Add("Item 2")
         oItems:Add("Item 3")
         oItems:Add("Item 4")
         oItems:Add("Item 5")
         oItems:Add("Item 6")
         oItems:Add("Item 7")
         oItems:Add("Item 8")
      oRadialMenu:EndUpdate()

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

method Items.Clear ()

Removes all objects in a collection.

Type	Description

The Clear method clears all items from the collection. The Count property specifies the number of items in the collection. The Item property accesses an item based on its index or name. The Remove method removes an item from the collection.

property Items.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A Long expression that specifies the number of items in the collection.

The Count property specifies the number of items in the collection. The Clear method clears all items from the collection. The Item property accesses an item based on its index or name. The Remove method removes an item from the collection.

property Items.Item (Index as Variant) as Item

Returns a specific Item object giving its index or name.

Type	Description
Index as Variant	A Long expression that specifies the index of the item to be requested, ro a string expression that specifies the caption/name of the item to be queried.
Item	An Item object being queried.

The Item property accesses an item based on its index or name. The Clear method clears all items from the collection. The Count property specifies the number of items in the collection. The Remove method removes an item from the collection.

method Items.Remove (Index as Variant)

Removes a specific member from the collection.

Type	Description
Index as Variant	A Long expression that specifies the index of the item to be requested, ro a string expression that specifies the caption/name of the item to be queried.

The Remove method removes an item from the collection. The Clear method clears all items from the collection. The Count property specifies the number of items in the collection. The Item property accesses an item based on its index or name.

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

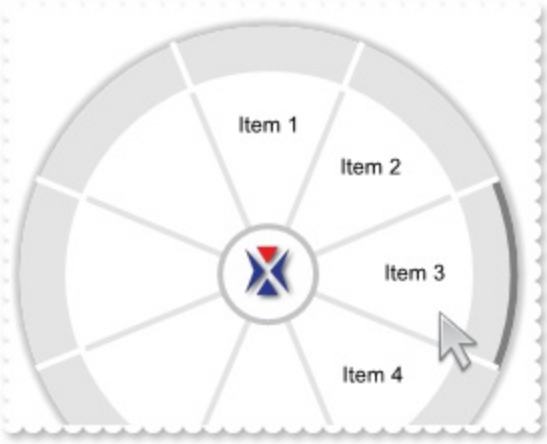
The ToString property loads or saves the control items from a string.

The user can add new items to the control using any of the following:

- Add method, adds a new item to the control. The Add method can be used to add child-items as well.
- ToString property of the Items collection, loads or saves the Items collection using string representation.
- ToString property of the control, loads or saves the Items collection using string representation.

The Remove method removes a specified item. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

For instance, the *"Item 1,Item 2,Item 3, Item 4"*, generates the following screen shot:



For instance, the *"Item 1,Item 2[scap=<font ;6>continue],Item 3 (Child 1,Child 2)"*, 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> ::= "scap" | "img" | "simg" | "bg" | "sbg" | "bga" | "sbga" | "fg" | "sfg" | "ttp" | "sttp" | "ttpt" | "sttpt" | "data" | "sdata" | "browse" | "custom" | "value"

where the <CAPTION> is the HTML caption to be shown on the item, equivalent with Caption(exRadialItems). The <VALUE> indicates the value of giving property.

- 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. This option is equivalent with the BackColor(exRadialItems) property.
- sbg=<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. This option is equivalent with the BackColor(exRadialSubItems) property.
- bga=<VALUE>, specifies the value of alpha / opacity channel to show the item's background color, where <VALUE> is a BYTE value. This option is equivalent with the BackAlpha(exRadialItems) property.
- sbga=<VALUE>, specifies the value of alpha / opacity channel to show the item's background color, where <VALUE> is a BYTE value. This option is equivalent with the BackAlpha(exRadialSubItems) property.
- browse=<VALUE>, specifies what the item displays, when the user clicks/browses it, where <VALUE> is a BrowseItemEnum value (0, 1 or 2). This option is equivalent with the BrowseType property.
- custom=<VALUE>, indicates the custom object to be shown when the user clicks/browses the item, where <VALUE> is a RadialCustomTypeEnum value (0, 16 or 32). This option is equivalent with the BrowseCustomType property.
- data=<VALUE>, indicates the item's user data, where <VALUE> is any expression.

This option is equivalent with the UserData(exRadialItems) property.

- sdata=<VALUE>, indicates the item's user data, where <VALUE> is any expression. This option is equivalent with the UserData(exRadialSubItems) property.
- fg=<VALUE>, indicates the item's foreground color, where <VALUE> is 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). This option is equivalent with the ForeColor(exRadialItems) property.
- sfg=<VALUE>, indicates the item's foreground color, where <VALUE> is 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). This option is equivalent with the ForeColor(exRadialSubItems) property.
- img=<VALUE>, indicates the item's image, where <VALUE> is name of the picture, key, icon, and so on. This option is equivalent with the Image(exRadialItems) property.
- simg=<VALUE>, indicates the item's image, where <VALUE> is name of the picture, key, icon, and so on. This option is equivalent with the Image(exRadialSubItems) property.
- scap=<VALUE>, indicates the item's caption, where <VALUE> is HTML text to be shown on the sub-item zone of the item. This option is equivalent with the Caption(exRadialSubItems) property.
- ttp=<VALUE>, indicates the item's tooltip, where <VALUE> is HTML text to be shown when the cursor hovers the item. This option is equivalent with the Tooltip(exRadialItems) property.
- sttp=<VALUE>, indicates the item's tooltip, where <VALUE> is HTML text to be shown when the cursor hovers the item. This option is equivalent with the Tooltip(exRadialSubItems) property.
- ttpt=<VALUE>, indicates the title of the item's tooltip, where <VALUE> is the title of the item's tooltip. This option is equivalent with the TooltipTitle(exRadialItems) property.
- sttpt=<VALUE>, indicates the title of the item's tooltip, where <VALUE> is the title of the item's tooltip. This option is equivalent with the TooltipTitle(exRadialSubItems) property.
- value=<VALUE>, indicates the item's value, where <VALUE> is the value. This option is equivalent with the BrowseCustom(exRadialCustomSliderValue) property.

RadialMenu object

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {1604BDE1-D48F-4D3F-B51B-49C0CD74236C}. The object's program identifier is: "Exontrol.RadialMenu". The /COM object module is: "ExRadialMenu.dll"

The eXRadialMenu (radial or pie menu) component is similar to the Microsoft's OneNote radial menu with ability to customize the appearance and functionality. The component is designed using tree structure so an item can hold none or more children, and so any item can be browsed, and show its children around it. An item can display a collection of child items, as well as a radial slider, or any other gauge / knob control. The eXRadialMenu is written from scratch, and does not depend on Windows 7, 8, 10 and so requires no dependencies to any other third party library.

The RadialMenu object supports the following properties and methods:

Name	Description
AllowBrowseItem	Specifies that the a new item gets browsed once the user clicks item.
AllowHotPointer	Indicates whether the pointer is oriented to the item, while hovering the radial menu.
AllowMoveOnFloat	Allows moving the control to a new position, when the user clicks and drags it over the screen, while it is floating (Float property is set to a non-zero value).
AllowToggleExpand	Specifies whether the radial menu can be shown in collapsed state.

AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
ArrowImage	Specifies the graphics (image, icon, picture) to be shown on the sub-items zone, for items that contains child items or sub items.
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.
BackgroundPicture	Indicates the picture to be shown on the radial menu's background.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BrowseItem	Specifies the item being browsed.
Caption	Specifies the caption on the control.
CustomBackAlpha	Specifies the value of alpha / opacity channel to show the custom portion of the radial menu.
CustomBackColor	Specifies the color to show the custom portion of the radial menu.
CustomHeight	Gets a value that represents the height of the inner custom control.
CustomLeft	Gets a value that represents the distance between the left side of the inner custom control and the left side of the control itself.
CustomPicture	Indicates the picture to be shown on the custom's background.
CustomTop	Gets a value that represents the distance between the left side of the inner custom control and the left side of the control itself.
CustomWidth	Gets a value that represents the width of the inner custom control.
DisplayAngle	Specifies the angle to display the items around the radial menu.
DisplayArrow	Indicates where the arrow of items with children is

	displayed.
DisplayCenter	Specifies the ratio to determine where the image/caption of the item is displayed.
DisplayCenterArrow	Specifies the ratio to determine where the arrow of items with children is displayed.
DisplayRadial	Determines how the item is displayed on the radial menu.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExcludeParentFromItems	Gets or sets a value that specifies whether the parent portion of the control is excluded from the items zone.
ExecuteTemplate	Executes a template and returns the result.
Expanded	Indicates whether the radial menu is expanded or collapsed.
ExtraCaption	Specifies any extra caption on the control.
Float	Specifies whether the control is shown as float.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
GoBack	Advances to the parent item.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Images List Control.
ImageSize	Retrieves or sets the size of icons the control displays..
IndexFromPoint	Retrieves the index of the radial pie, from the point.
InflateCustom	Inflates or deflates the client area of the custom portion of the control.
InflateItems	Inflates or deflates the client area of the items portion of

	the control.
InflateParentPicture	Inflates or deflates the client area to display the picture on the background of the parent's zone of the control.
InflateRadialMenu	Inflates or deflates the client area of the radial menu control.
ItemFromPoint	Retrieves the item, from the point.
Items	Retrieves the control's Items collection.
ItemsBackAlpha	Specifies the value of alpha / opacity channel to show the items portion of the radial menu.
ItemsBackColor	Specifies the color to show the items portion of the radial menu.
ItemsImageHeight	Specifies the height to display the item's image.
ItemsImageWidth	Specifies the width to display the item's image.
ItemsPicture	Indicates the picture to be shown on the items's background.
LayerUpdate	Specifies where the control updates its content.
MinVisibleCount	Specifies the minimum number of items being visible on the radial menu.
ParentBackAlpha	Specifies the value of alpha / opacity channel to show the items portion of the radial menu.
ParentBackColor	Specifies the color to show the parent portion of the radial menu.
ParentCaption	Specifies the caption to be shown on the parent zone, based on the state of the radial menu.
ParentImage	Specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu.
ParentImageHeight	Specifies the height to display the parent image in specified state.
ParentImageWidth	Specifies the width to display the parent image in specified state.
ParentOnPoint	Indicates if the point hits the parent zone of the radial menu.
ParentPicture	Indicates the picture to be shown on the parent zone's background.
ParentSize	Specifies the size to display the parent zone.

PicturesPath	Specifies the path to load the pictures from.
PointerAngle	Specifies the angle of the pointer to target another item or index.
PointerIndex	Specifies the index within the radial menu to target the pointer.
PointerPicture	Indicates the picture to be shown on the pointer zone's background.
PointerPictureHeight	Specifies the height of the the pointer, relative to the center of the radial menu.
PointerPictureWidth	Specifies the width of the the pointer, relative to the center of the radial menu.
PointerPictureX	Specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
PointerPictureY	Specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
RadialLineAlpha	Specifies the value of alpha / opacity channel to show the giving line within the radial menu.
RadialLineColor	Specifies the color to show the given radial line within the control.
RadialLineSize	Specifies the size to show the giving line within the radial menu.
RadialLineStyle	Specifies the style to show the given radial line within the control.
Refresh	Refreses the control.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
Root	Retrieves the root item.
SelBackAlpha	Specifies the value of alpha / opacity channel to show the selection of the radial menu.
SelBackColor	Specifies the selection background color.
SelectedIndex	Gets or sets a value that indicates index to be selected.
SelForeColor	Specifies the selection foreground color.
ShadowColor	Specifies the control's shadow color.
ShowImageList	Specifies whether the control's image list window is visible or hidden.

ShowToolTip	Shows the specified tooltip at given position.
State	Specifies the state of the radial menu.
SubItemsBackAlpha	Specifies the value of alpha / opacity channel to show the sub items zone of the radial menu.
SubItemsBackColor	Specifies the color to show the sub items zone of the radial menu.
SubItemsSize	Specifies the size to display the sub-items zone.
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.
ToTemplate	Generates the control's template.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.

property RadialMenu.AllowBrowseItem as Boolean

Specifies that the a new item gets browsed once the user clicks item.

Type	Description
Boolean	A Boolean expression that specifies that the a new item gets browsed once the user clicks item.

By default, the AllowBrowseItem property is True. The AllowBrowseItem property specifies that the a new item gets browsed once the user clicks item. For instance, you can use the AllowBrowseItem property on False, to disable browsing for new items when user clicks an item / parent item. The BrowseItem property specifies the item currently browsed. The SelectItem event notifies once the user selects an item. The SelectParent event occurs once the user clicks the parent of the item.

property RadialMenu.AllowHotPointer as Boolean

Indicates whether the pointer is oriented to the item, while hovering the radial menu.

Type	Description
Boolean	A Boolean expression that indicates whether the pointer is oriented to the item, while hovering the radial menu.

By default, the AllowHotPointer property is False. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The SelectedIndex property specifies the index of the item/slice to be selected.

The following screen show show a pointer over the control:

The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

The PointerIndex property specifies the index within the radial menu to target the pointer.
The PointerAngle property specifies the angle of the pointer to target another item or index.

The following sample show how you can assign a hot pointer to your radial menu:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .PointerPicture = "pointer.png"
   .PointerIndex = -1
   .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
   .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
   .AllowHotPointer = True
   .SelBackAlpha(3) = 128
   .SelForeColor(3) = RGB(0,0,0)
   .ParentSize = "36*dpi"
   .ParentImageHeight(-1) = "48*dpi"
   .ParentImageWidth(-1) = "48*dpi"
   .RadialLineSize(8) = -1
   .RadialLineAlpha(8) = 32
   .RadialLineColor(11) = -1
   .Expanded = True
   .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
   .EndUpdate
End With
```

VB6

```
With RadialMenu1
   .BeginUpdate
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .PointerPicture = "pointer.png"
   .PointerIndex = -1
   .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
   .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
   .AllowHotPointer = True
   .SelBackAlpha(exRadialFullItems) = 128
```

```
    .SelForeColor(exRadialFullItems) = RGB(0,0,0)
    .ParentSize = "36*dpi"
    .ParentImageHeight(exRadialMenuStateAll) = "48*dpi"
    .ParentImageWidth(exRadialMenuStateAll) = "48*dpi"
    .RadialLineSize(exRadialHotParent) = -1
    .RadialLineAlpha(exRadialHotParent) = 32
    .RadialLineColor(exRadialHotFullItem) = -1
    .Expanded = True
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .PointerPicture = "pointer.png"
    .PointerIndex = -1
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = True

.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,128


.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,Colo

    .ParentSize = "36*dpi"

.set_ParentImageHeight(exontrol.EXRADIALMENULib.RadialMenuStateEnum.exRadialM


.set_ParentImageWidth(exontrol.EXRADIALMENULib.RadialMenuStateEnum.exRadialM


.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)
```

```
.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,3

.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotFullIte

    .Expanded = True
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
    .BeginUpdate()
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .PointerPicture = "pointer.png"
    .PointerIndex = -1
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = True
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,128)
    .set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)
    .ParentSize = "36*dpi"

.set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState



.set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState/

    .set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)
    .set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32)
    .set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,-1)
    .Expanded = True
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .EndUpdate()
```

```
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutPicturesPath(L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
spRadialMenu1->PutPointerPicture("pointer.png");
spRadialMenu1->PutPointerIndex(-1);
spRadialMenu1->PutPointerPictureY(L"y + (height-pheight)/2- 21*dpi");
spRadialMenu1->PutPointerPictureX(L"x + (width-pwidth)/2 + 1 * dpi");
spRadialMenu1->PutAllowHotPointer(VARIANT_TRUE);
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialFullItems,128);
spRadialMenu1->PutSelForeColor(EXRADIALMENULib::exRadialFullItems,RGB(0,0,0));
spRadialMenu1->PutParentSize(L"36*dpi");
spRadialMenu1-
>PutParentImageHeight(EXRADIALMENULib::exRadialMenuStateAll,L"48*dpi");
spRadialMenu1-
>PutParentImageWidth(EXRADIALMENULib::exRadialMenuStateAll,L"48*dpi");
spRadialMenu1->PutRadialLineSize(EXRADIALMENULib::exRadialHotParent,-1);
spRadialMenu1->PutRadialLineAlpha(EXRADIALMENULib::exRadialHotParent,32);
spRadialMenu1->PutRadialLineColor(EXRADIALMENULib::exRadialHotFullItem,-1);
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->GetItems()->PutToString(L"Item 1(1),Item 2,Item 3(1),Item 4,Item
5,Item 6,Item 7,Item 8");
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->PicturesPath = L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
RadialMenu1->set_PointerPicture(TVariant("pointer.png"));
RadialMenu1->PointerIndex = -1;
RadialMenu1->PointerPictureY = L"y + (height-pheight)/2- 21*dpi";
RadialMenu1->PointerPictureX = L"x + (width-pwidth)/2 + 1 * dpi";
RadialMenu1->AllowHotPointer = true;
RadialMenu1-
>SelBackAlpha[ExradialmenulibtlbRadialItemsEnumexRadialFullItems] = 128;
RadialMenu1-
>SelForeColor[ExradialmenulibtlbRadialItemsEnumexRadialFullItems] =
RGB(0,0,0);
RadialMenu1->ParentSize = L"36*dpi";
RadialMenu1-
>ParentImageHeight[ExradialmenulibtlbRadialMenuStateEnumexRadialMenuStateA
 = L"48*dpi";
RadialMenu1-
>ParentImageWidth[ExradialmenulibtlbRadialMenuStateEnumexRadialMenuStateAl
 = L"48*dpi";
RadialMenu1-
>RadialLineSize[ExradialmenulibtlbRadialLineEnumexRadialHotParent] = -1;
RadialMenu1-
>RadialLineAlpha[ExradialmenulibtlbRadialLineEnumexRadialHotParent] = 32;
RadialMenu1-
>RadialLineColor[ExradialmenulibtlbRadialLineEnumexRadialHotFullItem] = -1;
RadialMenu1->Expanded = true;
RadialMenu1->Items->ToString = L"Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8";
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.PicturesPath = "C:\\Program
```

```
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
exradialmenu1.PointerPicture = "pointer.png";
exradialmenu1.PointerIndex = -1;
exradialmenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
exradialmenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
exradialmenu1.AllowHotPointer = true;
exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadia

exradialmenu1.ParentSize = "36*dpi";
exradialmenu1.set_ParentImageHeight(exontrol.EXRADIALMENULib.RadialMenuStateE

exradialmenu1.set_ParentImageWidth(exontrol.EXRADIALMENULib.RadialMenuStateEr

exradialmenu1.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadia

exradialmenu1.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRad

exradialmenu1.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exR

exradialmenu1.Expanded = true;
exradialmenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
```

```
   RadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
   RadialMenu1.PointerPicture = "pointer.png";
   RadialMenu1.PointerIndex = -1;
   RadialMenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
   RadialMenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
   RadialMenu1.AllowHotPointer = true;
   RadialMenu1.SelBackAlpha(3) = 128;
   RadialMenu1.SelForeColor(3) = 0;
   RadialMenu1.ParentSize = "36*dpi";
   RadialMenu1.ParentImageHeight(-1) = "48*dpi";
   RadialMenu1.ParentImageWidth(-1) = "48*dpi";
   RadialMenu1.RadialLineSize(8) = -1;
   RadialMenu1.RadialLineAlpha(8) = 32;
   RadialMenu1.RadialLineColor(11) = -1;
   RadialMenu1.Expanded = true;
   RadialMenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
      .PointerPicture = "pointer.png"
      .PointerIndex = -1
```

```
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = True
    .SelBackAlpha(3) = 128
    .SelForeColor(3) = RGB(0,0,0)
    .ParentSize = "36*dpi"
    .ParentImageHeight(-1) = "48*dpi"
    .ParentImageWidth(-1) = "48*dpi"
    .RadialLineSize(8) = -1
    .RadialLineAlpha(8) = 32
    .RadialLineColor(11) = -1
    .Expanded = True
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .EndUpdate
  End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
axRadialMenu1.PointerPicture = "pointer.png";
axRadialMenu1.PointerIndex = -1;
axRadialMenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
axRadialMenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
axRadialMenu1.AllowHotPointer = true;
axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialFullIte

axRadialMenu1.set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullIter
(uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0)));
axRadialMenu1.ParentSize = "36*dpi";
axRadialMenu1.set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exR
```

```
axRadialMenu1.set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRa

axRadialMenu1.set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotPar

axRadialMenu1.set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotP

axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFu

axRadialMenu1.Expanded = true;
axRadialMenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    ;

    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.PicturesPath("C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
    exradialmenu1.PointerPicture("pointer.png");
    exradialmenu1.PointerIndex(-1);
    exradialmenu1.PointerPictureY("y + (height-pheight)/2- 21*dpi");
    exradialmenu1.PointerPictureX("x + (width-pwidth)/2 + 1 * dpi");
    exradialmenu1.AllowHotPointer(true);
    exradialmenu1.SelBackAlpha(3/*exRadialFullItems*/,128);
    exradialmenu1.SelForeColor(3/*exRadialFullItems*/,WinApi::RGB2int(0,0,0));
    exradialmenu1.ParentSize("36*dpi");
    exradialmenu1.ParentImageHeight(-1/*exRadialMenuStateAll*/,"48*dpi");
    exradialmenu1.ParentImageWidth(-1/*exRadialMenuStateAll*/,"48*dpi");
    exradialmenu1.RadialLineSize(8/*exRadialHotParent*/,-1);
    exradialmenu1.RadialLineAlpha(8/*exRadialHotParent*/,32);
```

```
    exradialmenu1.RadialLineColor(11/*exRadialHotFullItem*/,-1);
    exradialmenu1.Expanded(true);
    exradialmenu1.Items().ToString("Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8");
    exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
    BeginUpdate();
    PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
    PointerPicture := 'pointer.png';
    PointerIndex := -1;
    PointerPictureY := 'y + (height-pheight)/2- 21*dpi';
    PointerPictureX := 'x + (width-pwidth)/2 + 1 * dpi';
    AllowHotPointer := True;
    set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,128);
    set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,$0);
    ParentSize := '36*dpi';

set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState



set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuStateA

    set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1);
    set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32);

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,$ffffffff);

    Expanded := True;
    Items.ToString := 'Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8';
    EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
  BeginUpdate();
  PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
  PointerPicture := 'pointer.png';
  PointerIndex := -1;
  PointerPictureY := 'y + (height-pheight)/2- 21*dpi';
  PointerPictureX := 'x + (width-pwidth)/2 + 1 * dpi';
  AllowHotPointer := True;
  SelBackAlpha[EXRADIALMENULib_TLB.exRadialFullItems] := 128;
  SelForeColor[EXRADIALMENULib_TLB.exRadialFullItems] := $0;
  ParentSize := '36*dpi';
  ParentImageHeight[EXRADIALMENULib_TLB.exRadialMenuStateAll] := '48*dpi';
  ParentImageWidth[EXRADIALMENULib_TLB.exRadialMenuStateAll] := '48*dpi';
  RadialLineSize[EXRADIALMENULib_TLB.exRadialHotParent] := -1;
  RadialLineAlpha[EXRADIALMENULib_TLB.exRadialHotParent] := 32;
  RadialLineColor[EXRADIALMENULib_TLB.exRadialHotFullItem] := $ffffffff;
  Expanded := True;
  Items.ToString := 'Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8';
  EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
  .BeginUpdate
  .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
  .PointerPicture = "pointer.png"
  .PointerIndex = -1
  .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
  .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
  .AllowHotPointer = .T.
  .Object.SelBackAlpha(3) = 128
  .Object.SelForeColor(3) = RGB(0,0,0)
  .ParentSize = "36*dpi"
  .Object.ParentImageHeight(-1) = "48*dpi"
  .Object.ParentImageWidth(-1) = "48*dpi"
```

```
   .Object.RadialLineSize(8) = -1
   .Object.RadialLineAlpha(8) = 32
   .Object.RadialLineColor(11) = -1
   .Expanded = .T.
   .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
   .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.PointerIndex = -1
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = true
oRadialMenu.Template = [SelBackAlpha(3) = 128] // oRadialMenu.SelBackAlpha(3) =
128
oRadialMenu.Template = [SelForeColor(3) = 0] // oRadialMenu.SelForeColor(3) = 0x0
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.Template = [ParentImageHeight(-1) = "48*dpi"] //
oRadialMenu.ParentImageHeight(-1) = "48*dpi"
oRadialMenu.Template = [ParentImageWidth(-1) = "48*dpi"] //
oRadialMenu.ParentImageWidth(-1) = "48*dpi"
oRadialMenu.Template = [RadialLineSize(8) = -1] // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = [RadialLineAlpha(8) = 32] //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = [RadialLineColor(11) = -1] //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
```

```
7,Item 8"
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.PointerIndex = -1
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = .t.
oRadialMenu.Template = "SelBackAlpha(3) = 128" // oRadialMenu.SelBackAlpha(3) =
128
oRadialMenu.Template = "SelForeColor(3) = 0" // oRadialMenu.SelForeColor(3) = 0
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.Template = "ParentImageHeight(-1) = `48*dpi`" //
oRadialMenu.ParentImageHeight(-1) = "48*dpi"
oRadialMenu.Template = "ParentImageWidth(-1) = `48*dpi`" //
oRadialMenu.ParentImageWidth(-1) = "48*dpi"
oRadialMenu.Template = "RadialLineSize(8) = -1" // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = "RadialLineAlpha(8) = 32" //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = "RadialLineColor(11) = -1" //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = .t.
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8"
oRadialMenu.EndUpdate()
```

Visual Objects

```
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:PicturesPath := "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
oDCOCX_Exontrol1:PointerPicture := "pointer.png"
oDCOCX_Exontrol1:PointerIndex := -1
oDCOCX_Exontrol1:PointerPictureY := "y + (height-pheight)/2- 21*dpi"
oDCOCX_Exontrol1:PointerPictureX := "x + (width-pwidth)/2 + 1 * dpi"
oDCOCX_Exontrol1:AllowHotPointer := true
oDCOCX_Exontrol1:[SelBackAlpha,exRadialFullItems] := 128
oDCOCX_Exontrol1:[SelForeColor,exRadialFullItems] := RGB(0,0,0)
oDCOCX_Exontrol1:ParentSize := "36*dpi"
oDCOCX_Exontrol1:[ParentImageHeight,exRadialMenuStateAll] := "48*dpi"
oDCOCX_Exontrol1:[ParentImageWidth,exRadialMenuStateAll] := "48*dpi"
oDCOCX_Exontrol1:[RadialLineSize,exRadialHotParent] := -1
oDCOCX_Exontrol1:[RadialLineAlpha,exRadialHotParent] := 32
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotFullItem] := -1
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:Items:ToString := "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.PointerIndex = -1
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = true
oRadialMenu.SelBackAlpha(3,128)
```

```
oRadialMenu.SelForeColor(3,RGB(0,0,0))
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.ParentImageHeight(-1,"48*dpi")
oRadialMenu.ParentImageWidth(-1,"48*dpi")
oRadialMenu.RadialLineSize(8,-1)
oRadialMenu.RadialLineAlpha(8,32)
oRadialMenu.RadialLineColor(11,-1)
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
    Forward Send OnCreate
    Send ComBeginUpdate
    Set ComPicturesPath to "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    Set ComPointerPicture to "pointer.png"
    Set ComPointerIndex to -1
    Set ComPointerPictureY to "y + (height-pheight)/2- 21*dpi"
    Set ComPointerPictureX to "x + (width-pwidth)/2 + 1 * dpi"
    Set ComAllowHotPointer to True
    Set ComSelBackAlpha OLEexRadialFullItems to 128
    Set ComSelForeColor OLEexRadialFullItems to (RGB(0,0,0))
    Set ComParentSize to "36*dpi"
    Set ComParentImageHeight OLEexRadialMenuStateAll to "48*dpi"
    Set ComParentImageWidth OLEexRadialMenuStateAll to "48*dpi"
    Set ComRadialLineSize OLEexRadialHotParent to -1
    Set ComRadialLineAlpha OLEexRadialHotParent to 32
    Set ComRadialLineColor OLEexRadialHotFullItem to -1
    Set ComExpanded to True
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
```

```
    Set pvComObject of hoItems to voItems
        Set ComToString of hoItems to "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8"
    Send Destroy to hoItems
    Send ComEndUpdate
End_Procedure
```

XBase++

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

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

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

    oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
    oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
    oRadialMenu:create(,, {10,60},{610,370} )

        oRadialMenu:BeginUpdate()
        oRadialMenu:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
        oRadialMenu:PointerPicture := "pointer.png"
        oRadialMenu:PointerIndex := -1
        oRadialMenu:PointerPictureY := "y + (height-pheight)/2- 21*dpi"
        oRadialMenu:PointerPictureX := "x + (width-pwidth)/2 + 1 * dpi"
        oRadialMenu:AllowHotPointer := .T.
        oRadialMenu:SetProperty("SelBackAlpha",3/*exRadialFullItems*/,128)
```

```
oRadialMenu:SetProperty("SelForeColor",3/*exRadialFullItems*/,AutomationTranslateC
 GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oRadialMenu:ParentSize := "36*dpi"

oRadialMenu:SetProperty("ParentImageHeight",-1/*exRadialMenuStateAll*/,"48*dpi")

oRadialMenu:SetProperty("ParentImageWidth",-1/*exRadialMenuStateAll*/,"48*dpi")
      oRadialMenu:SetProperty("RadialLineSize",8/*exRadialHotParent*/,-1)
      oRadialMenu:SetProperty("RadialLineAlpha",8/*exRadialHotParent*/,32)
      oRadialMenu:SetProperty("RadialLineColor",11/*exRadialHotFullItem*/,-1)
      oRadialMenu:Expanded := .T.
      oRadialMenu:Items():ToString := "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8"
      oRadialMenu:EndUpdate()

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

property RadialMenu.AllowMoveOnFloat as Boolean

Allows moving the control to a new position, when the user clicks and drags it over the screen, while it is floating (Float property is set to a non-zero value).

Type	Description
Boolean	A Boolean expression that specifies whether the control can be moved by dragging.

By default, the AllowMoveOnFloat property is True, which allows moving the control to a new position, when the user clicks and drags it over the screen, while it is floating (Float property is set to a non-zero value). The AllowMoveOnFloat property has effect only if the control's Float property is exRadialMenuFloat or exRadialMenuFloatTopmost. The Float property specifies whether the control is shown as float.

property RadialMenu.AllowToggleExpand as Boolean

Specifies whether the radial menu can be shown in collapsed state.

Type	Description
Boolean	A Boolean expression that specifies whether the radial menu can be shown in collapsed state.

By default, the AllowToggleExpand property is True, which indicates that the user can expand or collapse the radial menu. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu. The ParentCaption property specifies the caption to be displayed on the parent portion of the control, based on the radial menu's state. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state.

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

Retrieves the identifier of the anchor from point.

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

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

property RadialMenu.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the control's appearance, or a color expression whose last 7 bits in the high significant byte of the value indicates the index of the skin in the Appearance collection, being displayed as control's borders. For instance, if the Appearance = 0x1000000, indicates that the first skin object in the Appearance collection defines the control's border. The Client object in the skin, defines the client area of the control.

Use the Appearance property to specify the control's border.

property RadialMenu.ArrowImage as Variant

Specifies the graphics (image, icon, picture) to be shown on the sub-items zone, for items that contains child items or sub items.

Type	Description
Variant	A VARIANT expression that specifies the icon/picture/image to be displayed as described: • A String expression indicates: ◦ a name of a picture file in the PicturePath folder. For instance, ArrowImage = "favorites.png", loads the favorites.png file if found in the PicturePath folder. ◦ a picture file including its absolute path. For instance, ArrowImage = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favo loads the favorites.png file from absolute path ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, ArrowImage = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favo ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on. • A Picture object that indicates the picture to be displayed. For instance, ArrowImage = LoadPicture("picture.jpg") • a long/string expression that specifies the index of the icon to be displayed (0-based). The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. If no icon/picture/image is found, the object displays no icon/picture/image.

By default, the ArrowImage property is "arrow" (^). The ArrowImage property specifies

the graphics (image, icon, picture) to be shown on the sub-items zone, for items that contains child items or sub items. Use the Add / ToString method to add new items to the control. The RadialLineColor(exRadialItemsChildren) specifies the color to show the items with children, in the items portion of the control. The RadialLineColor(exRadialSubItemsChildren) specifies the color to show the items with children, in the sub-items portion of the control.

The following screen shot shows the items with children with a different color and a different arrow:

method RadialMenu.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 RadialMenu1_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.

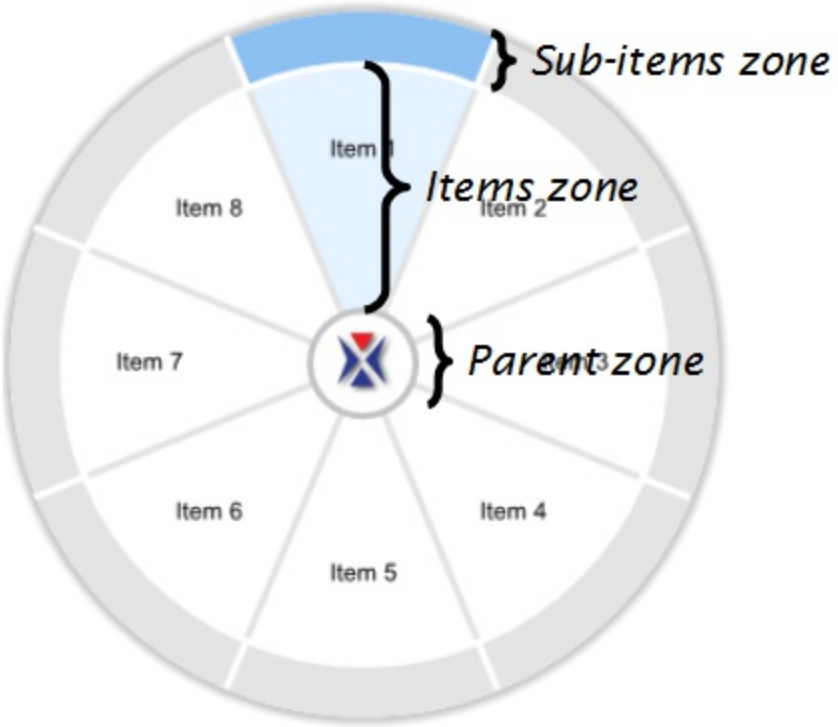
property RadialMenu.BackColor as Color

Specifies the control's background color.

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

The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu. The BackgroundPicture property indicates the picture to be shown on the radial menu's background.

The following screen shot, shows the portions/parts/zones of the radial menu:



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

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

For instance:

- Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips
- Use the Background(exToolTipBackColor) property indicates the tooltip's background color
- Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

Use the ShowToolTip method to display a custom tooltip. The Tooltip / TooltipTitle property indicates the item's tooltip.

property RadialMenu.BackgroundPicture as Variant

Indicates the picture to be shown on the radial menu's background.

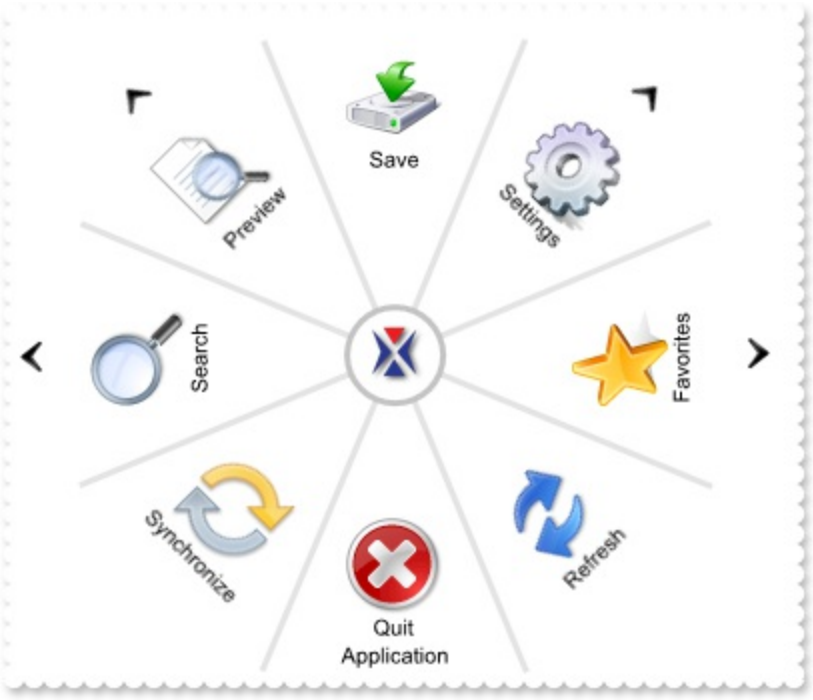
Type	Description

Variant	A VARIANT expression that specifies the icon/picture/image to be displayed as described:
	
	• A String expression indicates:
	  ◦ a name of a picture file in the PicturePath folder. For instance, BackgroundPicture = "favorites.png", loads the favorites.png file if found in the PicturePath folder.
	  ◦ a picture file including its absolute path. For instance, BackgroundPicture = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor loads the favorites.png file from absolute path
	  ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, BackgroundPicture = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor
	  ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on.
	• A Picture object that indicates the picture to be displayed. For instance, BackgroundPicture = LoadPicture("picture.jpg")
	
	If no picture/image is found, the item displays no picture/image.

By default, The BackgroundPicture property is empty. The BackgroundPicture property indicates the picture to be shown on the radial menu's background. The ItemsPicture property indicates the picture to be shown on the items's background. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The SubItemsSize property specifies the size to display the

sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu.

The following screen shot shows control with no background picture (default):



The following screen shot shows control with a background picture:

method RadialMenu.BeginUpdate ()

Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.

Type	Description

property RadialMenu.BrowseItem as Item

Specifies the item being browsed.

Type	Description
Item	An Item object being browsed.

The BrowseItem property indicates the item being currently browsed. The AllowBrowseItem property specifies that the a new item gets browsed once the user clicks item. The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The Item property returns an Item based on its index or caption.

The following samples show how you can programmatically browses for a new item:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .Expanded = True
   .ShadowColor = -1
   .InflateItems = "-8*dpi"
   .InflateCustom = .InflateItems
   .ItemsBackColor = RGB(240,240,240)
   .RadialLineColor(4) = RGB(0,0,0)
   .RadialLineStyle(4) = 2
   .RadialLineSize(4) = 3
   With .Items
      With .Add("Slider")
         .BrowseType = 2
         .BrowseCustomType = 16
      End With
   End With
   .BrowseItem = .Items.Item("Slider")
End With
```

VB6

```
With RadialMenu1
   .Expanded = True
```

```
    .ShadowColor = -1
    .InflateItems = "-8*dpi"
    .InflateCustom = .InflateItems
    .ItemsBackColor = RGB(240,240,240)
    .RadialLineColor(exRadialCustomBorder) = RGB(0,0,0)
    .RadialLineStyle(exRadialCustomBorder) = exRadialLineDot
    .RadialLineSize(exRadialCustomBorder) = 3
    With .Items
        With .Add("Slider")
            .BrowseType = exBrowseItemCustom
            .BrowseCustomType = exRadialCustomSlider
        End With
    End With
    .BrowseItem = .Items.Item("Slider")
End With
```

VB.NET

```
With Exradialmenu1
    .Expanded = True
    .ShadowColor32 = -1
    .InflateItems = "-8*dpi"
    .InflateCustom = .InflateItems
    .ItemsBackColor = Color.FromArgb(240,240,240)

.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialCustomBorc



.set_RadialLineStyle(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialCustomBord



.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialCustomBorde

    With .Items
        With .Add("Slider")
            .BrowseType =
exontrol.EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom
```

```
        .BrowseCustomType =
exontrol.EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider
      End With
   End With
   .BrowseItem = .Items.Item("Slider")
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .Expanded = True
   .GetOcx().ShadowColor = -1
   .InflateItems = "-8*dpi"
   .InflateCustom = .InflateItems
   .ItemsBackColor = RGB(240,240,240)
   .set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialCustomBorder,0)

.set_RadialLineStyle(EXRADIALMENULib.RadialLineEnum.exRadialCustomBorder,EXRAD

   .set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialCustomBorder,3)
   With .Items
      With .Add("Slider")
         .BrowseType = EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom
         .BrowseCustomType =
EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider
      End With
   End With
   .BrowseItem = .Items.Item("Slider")
End With
```

C++

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

   #import <ExRadialMenu.dll>
```

```
   using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->PutShadowColor(-1);
spRadialMenu1->PutInflateItems(L"-8*dpi");
spRadialMenu1->PutInflateCustom(spRadialMenu1->GetInflateItems());
spRadialMenu1->PutItemsBackColor(RGB(240,240,240));
spRadialMenu1-
>PutRadialLineColor(EXRADIALMENULib::exRadialCustomBorder,RGB(0,0,0));
spRadialMenu1-
>PutRadialLineStyle(EXRADIALMENULib::exRadialCustomBorder,EXRADIALMENULib::ex

spRadialMenu1->PutRadialLineSize(EXRADIALMENULib::exRadialCustomBorder,3);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
   EXRADIALMENULib::IItemPtr var_Item = var_Items-
>Add(L"Slider",vtMissing,vtMissing);
      var_Item->PutBrowseType(EXRADIALMENULib::exBrowseItemCustom);
      var_Item->PutBrowseCustomType(EXRADIALMENULib::exRadialCustomSlider);
spRadialMenu1->PutBrowseItem(((EXRADIALMENULib::IItemPtr)(spRadialMenu1-
>GetItems()->GetItem("Slider"))));
```

C++ Builder

```
RadialMenu1->Expanded = true;
RadialMenu1->ShadowColor = -1;
RadialMenu1->InflateItems = L"-8*dpi";
RadialMenu1->InflateCustom = RadialMenu1->InflateItems;
RadialMenu1->ItemsBackColor = RGB(240,240,240);
RadialMenu1-
>RadialLineColor[Exradialmenulib_tlb::RadialLineEnum::exRadialCustomBorder] =
RGB(0,0,0);
RadialMenu1-
>RadialLineStyle[Exradialmenulib_tlb::RadialLineEnum::exRadialCustomBorder] =
Exradialmenulib_tlb::RadialLineStyleEnum::exRadialLineDot;
```

```
RadialMenu1->RadialLineSize[Exradialmenulib_tlb::RadialLineEnum::exRadialCustomBorder] = 3;
Exradialmenulib_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   Exradialmenulib_tlb::IItemPtr var_Item = var_Items->Add(L"Slider",TNoParam(),TNoParam());
      var_Item->BrowseType = Exradialmenulib_tlb::BrowseItemEnum::exBrowseItemCustom;
      var_Item->BrowseCustomType = Exradialmenulib_tlb::RadialCustomTypeEnum::exRadialCustomSlider;
RadialMenu1->BrowseItem = RadialMenu1->Items->get_Item(TVariant("Slider"));
```

C#

```
exradialmenu1.Expanded = true;
exradialmenu1.ShadowColor32 = -1;
exradialmenu1.InflateItems = "-8*dpi";
exradialmenu1.InflateCustom = exradialmenu1.InflateItems;
exradialmenu1.ItemsBackColor = Color.FromArgb(240,240,240);
exradialmenu1.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRac

exradialmenu1.set_RadialLineStyle(exontrol.EXRADIALMENULib.RadialLineEnum.exRadi

exradialmenu1.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadia

exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   exontrol.EXRADIALMENULib.Item var_Item = var_Items.Add("Slider",null,null);
      var_Item.BrowseType = exontrol.EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom;
      var_Item.BrowseCustomType = exontrol.EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider;
exradialmenu1.BrowseItem = (exradialmenu1.Items["Slider"] as exontrol.EXRADIALMENULib.Item);
```

JScript/JavaScript

```
<BODY onload="Init()">
```

```
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.Expanded = true;
   RadialMenu1.ShadowColor = -1;
   RadialMenu1.InflateItems = "-8*dpi";
   RadialMenu1.InflateCustom = RadialMenu1.InflateItems;
   RadialMenu1.ItemsBackColor = 15790320;
   RadialMenu1.RadialLineColor(4) = 0;
   RadialMenu1.RadialLineStyle(4) = 2;
   RadialMenu1.RadialLineSize(4) = 3;
   var var_Items = RadialMenu1.Items;
      var var_Item = var_Items.Add("Slider",null,null);
         var_Item.BrowseType = 2;
         var_Item.BrowseCustomType = 16;
   RadialMenu1.BrowseItem = RadialMenu1.Items.Item("Slider");
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .Expanded = True
      .ShadowColor = -1
      .InflateItems = "-8*dpi"
      .InflateCustom = .InflateItems
```

```
      .ItemsBackColor = RGB(240,240,240)
      .RadialLineColor(4) = RGB(0,0,0)
      .RadialLineStyle(4) = 2
      .RadialLineSize(4) = 3
      With .Items
         With .Add("Slider")
            .BrowseType = 2
            .BrowseCustomType = 16
         End With
      End With
      .BrowseItem = .Items.Item("Slider")
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.Expanded = true;
(axRadialMenu1.GetOcx() as EXRADIALMENULib.RadialMenu).ShadowColor = -1;
axRadialMenu1.InflateItems = "-8*dpi";
axRadialMenu1.InflateCustom = axRadialMenu1.InflateItems;
axRadialMenu1.ItemsBackColor = Color.FromArgb(240,240,240);
axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialCusto
(uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0)));
axRadialMenu1.set_RadialLineStyle(EXRADIALMENULib.RadialLineEnum.exRadialCusto

axRadialMenu1.set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialCustom

EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
   EXRADIALMENULib.Item var_Item = var_Items.Add("Slider",null,null);
      var_Item.BrowseType =
EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom;
      var_Item.BrowseCustomType =
EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider;
axRadialMenu1.BrowseItem = (axRadialMenu1.Items["Slider"] as
```

```
EXRADIALMENULib.Item);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Item,com_Items;
    anytype var_Item,var_Items;
    ;

    super();

    exradialmenu1.Expanded(true);
    exradialmenu1.ShadowColor(-1);
    exradialmenu1.InflateItems("-8*dpi");
    exradialmenu1.InflateCustom(exradialmenu1.InflateItems());
    exradialmenu1.ItemsBackColor(WinApi::RGB2int(240,240,240));

exradialmenu1.RadialLineColor(4/*exRadialCustomBorder*/,WinApi::RGB2int(0,0,0));
    exradialmenu1.RadialLineStyle(4/*exRadialCustomBorder*/,2/*exRadialLineDot*/);
    exradialmenu1.RadialLineSize(4/*exRadialCustomBorder*/,3);
    var_Items = exradialmenu1.Items(); com_Items = var_Items;
       var_Item = com_Items.Add("Slider"); com_Item = var_Item;
          com_Item.BrowseType(2/*exBrowseItemCustom*/);
          com_Item.BrowseCustomType(16/*exRadialCustomSlider*/);
    exradialmenu1.BrowseItem(exradialmenu1.Items().Item("Slider"));
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
    Expanded := True;
    (GetOcx() as EXRADIALMENULib.RadialMenu).ShadowColor := $ffffffff;
    InflateItems := '-8*dpi';
    InflateCustom := InflateItems;
    ItemsBackColor := Color.FromArgb(240,240,240);
```

```
set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialCustomBorder,$0);

set_RadialLineStyle(EXRADIALMENULib.RadialLineEnum.exRadialCustomBorder,EXRADI

  set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialCustomBorder,3);
  with Items do
  begin
    with Add('Slider',Nil,Nil) do
    begin
      BrowseType := EXRADIALMENULib.BrowseItemEnum.exBrowseItemCustom;
      BrowseCustomType :=
EXRADIALMENULib.RadialCustomTypeEnum.exRadialCustomSlider;
    end;
  end;
  BrowseItem := (Items.Item['Slider'] as EXRADIALMENULib.Item);
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
  Expanded := True;
  ShadowColor := $ffffffff;
  InflateItems := '-8*dpi';
  InflateCustom := InflateItems;
  ItemsBackColor := RGB(240,240,240);
  RadialLineColor[EXRADIALMENULib_TLB.exRadialCustomBorder] := $0;
  RadialLineStyle[EXRADIALMENULib_TLB.exRadialCustomBorder] :=
EXRADIALMENULib_TLB.exRadialLineDot;
  RadialLineSize[EXRADIALMENULib_TLB.exRadialCustomBorder] := 3;
  with Items do
  begin
    with Add('Slider',Null,Null) do
    begin
      BrowseType := EXRADIALMENULib_TLB.exBrowseItemCustom;
      BrowseCustomType := EXRADIALMENULib_TLB.exRadialCustomSlider;
```

```
      end;
   end;
   BrowseItem := (IUnknown(Items.Item['Slider']) as EXRADIALMENULib_TLB.Item);
end
```

VFP

```
with thisform.RadialMenu1
   .Expanded = .T.
   .ShadowColor = -1
   .InflateItems = "-8*dpi"
   .InflateCustom = .InflateItems
   .ItemsBackColor = RGB(240,240,240)
   .Object.RadialLineColor(4) = RGB(0,0,0)
   .Object.RadialLineStyle(4) = 2
   .Object.RadialLineSize(4) = 3
   with .Items
      with .Add("Slider")
         .BrowseType = 2
         .BrowseCustomType = 16
      endwith
   endwith
   .BrowseItem = .Items.Item("Slider")
endwith
```

dBASE Plus

```
local oRadialMenu,var_Item,var_Items

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.Expanded = true
oRadialMenu.ShadowColor = -1
oRadialMenu.InflateItems = "-8*dpi"
oRadialMenu.InflateCustom = oRadialMenu.InflateItems
oRadialMenu.ItemsBackColor = 0xf0f0f0
oRadialMenu.Template = [RadialLineColor(4) = 0] // oRadialMenu.RadialLineColor(4)
= 0x0
oRadialMenu.Template = [RadialLineStyle(4) = 2] // oRadialMenu.RadialLineStyle(4) =
```

```
2
oRadialMenu.Template = [RadialLineSize(4) = 3] // oRadialMenu.RadialLineSize(4) =
3
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("Slider")
      var_Item.BrowseType = 2
      var_Item.BrowseCustomType = 16
oRadialMenu.BrowseItem = oRadialMenu.Items.Item("Slider")
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Item as P
Dim var_Items as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.Expanded = .t.
oRadialMenu.ShadowColor = -1
oRadialMenu.InflateItems = "-8*dpi"
oRadialMenu.InflateCustom = oRadialMenu.InflateItems
oRadialMenu.ItemsBackColor = 15790320
oRadialMenu.Template = "RadialLineColor(4) = 0" // oRadialMenu.RadialLineColor(4)
= 0
oRadialMenu.Template = "RadialLineStyle(4) = 2" // oRadialMenu.RadialLineStyle(4)
= 2
oRadialMenu.Template = "RadialLineSize(4) = 3" // oRadialMenu.RadialLineSize(4) =
3
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("Slider")
      var_Item.BrowseType = 2
      var_Item.BrowseCustomType = 16
oRadialMenu.BrowseItem = oRadialMenu.Items.Item("Slider")
```

Visual Objects

```
local var_Item as IItem
```

```
local var_Items as IItems

oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:ShadowColor := -1
oDCOCX_Exontrol1:InflateItems := "-8*dpi"
oDCOCX_Exontrol1:InflateCustom := oDCOCX_Exontrol1:InflateItems
oDCOCX_Exontrol1:ItemsBackColor := RGB(240,240,240)
oDCOCX_Exontrol1:[RadialLineColor,exRadialCustomBorder] := RGB(0,0,0)
oDCOCX_Exontrol1:[RadialLineStyle,exRadialCustomBorder] := exRadialLineDot
oDCOCX_Exontrol1:[RadialLineSize,exRadialCustomBorder] := 3
var_Items := oDCOCX_Exontrol1:Items
   var_Item := var_Items:Add("Slider",nil,nil)
      var_Item:BrowseType := exBrowseItemCustom
      var_Item:BrowseCustomType := exRadialCustomSlider
oDCOCX_Exontrol1:BrowseItem := IItem{oDCOCX_Exontrol1:Items:[Item,"Slider"]}
```

PowerBuilder

```
OleObject oRadialMenu,var_Item,var_Items

oRadialMenu = ole_1.Object
oRadialMenu.Expanded = true
oRadialMenu.ShadowColor = -1
oRadialMenu.InflateItems = "-8*dpi"
oRadialMenu.InflateCustom = oRadialMenu.InflateItems
oRadialMenu.ItemsBackColor = RGB(240,240,240)
oRadialMenu.RadialLineColor(4,RGB(0,0,0))
oRadialMenu.RadialLineStyle(4,2)
oRadialMenu.RadialLineSize(4,3)
var_Items = oRadialMenu.Items
   var_Item = var_Items.Add("Slider")
      var_Item.BrowseType = 2
      var_Item.BrowseCustomType = 16
oRadialMenu.BrowseItem = oRadialMenu.Items.Item("Slider")
```

Visual DataFlex

```
Procedure OnCreate
    Forward Send OnCreate
    Set ComExpanded to True
    Set ComShadowColor to -1
    Set ComInflateItems to "-8*dpi"
    Set ComInflateCustom to (ComInflateItems(Self))
    Set ComItemsBackColor to (RGB(240,240,240))
    Set ComRadialLineColor OLEexRadialCustomBorder to (RGB(0,0,0))
    Set ComRadialLineStyle OLEexRadialCustomBorder to OLEexRadialLineDot
    Set ComRadialLineSize OLEexRadialCustomBorder to 3
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Variant voItem
        Get ComAdd of hoItems "Slider" Nothing Nothing to voItem
        Handle hoItem
        Get Create (RefClass(cComItem)) to hoItem
        Set pvComObject of hoItem to voItem
            Set ComBrowseType of hoItem to OLEexBrowseItemCustom
            Set ComBrowseCustomType of hoItem to OLEexRadialCustomSlider
        Send Destroy to hoItem
    Send Destroy to hoItems
    Variant v
    Variant voItems1
    Get ComItems to voItems1
    Handle hoItems1
    Get Create (RefClass(cComItems)) to hoItems1
    Set pvComObject of hoItems1 to voItems1
        Get ComItem of hoItems1 "Slider" to v
    Send Destroy to hoItems1
    Set ComBrowseItem to v
End_Procedure
```

XBase++

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

PROCEDURE Main
   LOCAL oForm
   LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
   LOCAL oItem
   LOCAL oItems
   LOCAL oRadialMenu

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:Expanded := .T.
      oRadialMenu:SetProperty("ShadowColor",-1)
      oRadialMenu:InflateItems := "-8*dpi"
      oRadialMenu:InflateCustom := oRadialMenu:InflateItems()
      oRadialMenu:SetProperty("ItemsBackColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 240,240,240 } )  , .F. ))

oRadialMenu:SetProperty("RadialLineColor",4/*exRadialCustomBorder*/,AutomationTr
 GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))

oRadialMenu:SetProperty("RadialLineStyle",4/*exRadialCustomBorder*/,2/*exRadialLin

      oRadialMenu:SetProperty("RadialLineSize",4/*exRadialCustomBorder*/,3)
      oItems := oRadialMenu:Items()
         oItem := oItems:Add("Slider")
            oItem:BrowseType := 2/*exBrowseItemCustom*/
            oItem:BrowseCustomType := 16/*exRadialCustomSlider*/
```

```
      oRadialMenu:BrowseItem := oRadialMenu:Items:Item("Slider")

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

property RadialMenu.Caption(Property as PropertyLayerCaptionEnum) as Variant

Specifies the caption on the control.

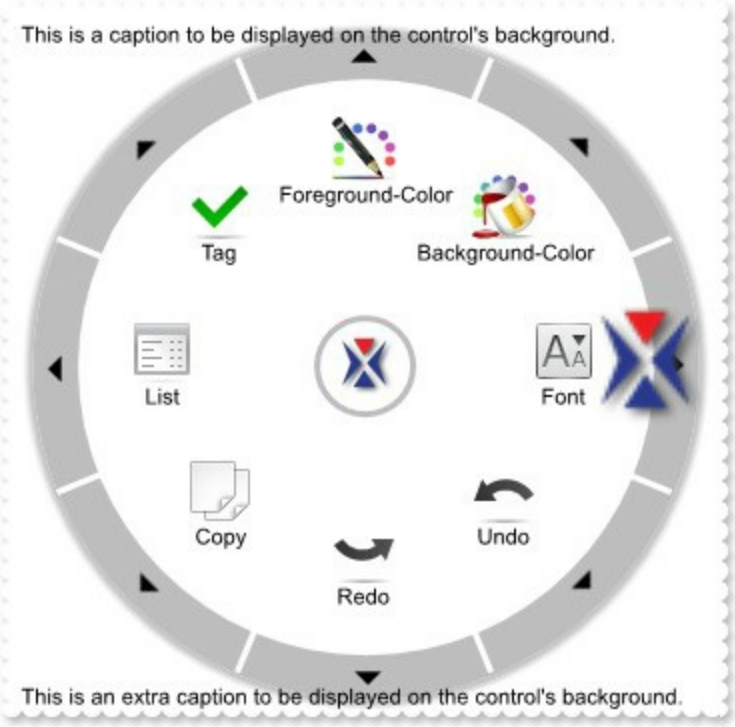
Type	Description
Property as PropertyLayerCaptionEnum	A PropertyLayerCaptionEnum expression that specifies the caption's property to be changed.
Variant	A VARIANT expression that specifies the value of the caption's property.

The control support unlimited HTML captions to be place anywhere on the control. The Caption(exLayerCaption) specifies the HTML caption to be shown on the control/layer. The Images method specifies the list of icons the control can display. The HTMLPicture adds or replaces a picture in HTML captions. The Caption(exLayerCaptionBackgroundExt) property indicates unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the control / layer's background. The caption on the control stay on its position, no matter what layer is moved or rotated, while a caption on a layer gets moved or rotated or clipped together with the layer itself. The ForeColor property specifies the control's foreground color.

Any of the following properties can be used to display a HTML caption:

- Caption property specifies the caption to be shown on the control's foreground.
- ExtraCaption property specifies any extra caption to be shown on the control's foreground.

The following screen shot shows a few captions on the control's background:



The following samples show how you can display captions on the control's background:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   .MinVisibleCount = 6
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
   .Caption(0) = "This is a caption to be displayed on the control's background."
   .ExtraCaption("extra",0) = "This is an extra caption to be displayed on the control's background."
   .ExtraCaption("extra",3) = 2
   .ExtraCaption("extra",8) = True
   .ExtraCaption("extralogo",0) = "<img>logo:64</img>"
   .ExtraCaption("extralogo",3) = 4
   .ExtraCaption("extralogo",4) = "width-twidth"
   .EndUpdate
End With
```

VB6

```
With RadialMenu1
   .BeginUpdate
```

```
    .Expanded = True
    .MinVisibleCount = 6
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
    .Caption(exLayerCaption) = "This is a caption to be displayed on the control's
background."
    .ExtraCaption("extra",exLayerCaption) = "This is an extra caption to be displayed
on the control's background."
    .ExtraCaption("extra",exLayerCaptionAnchor) = 2
    .ExtraCaption("extra",exLayerCaptionWordWrap) = True
    .ExtraCaption("extralogo",exLayerCaption) = "<img>logo:64</img>"
    .ExtraCaption("extralogo",exLayerCaptionAnchor) = 4
    .ExtraCaption("extralogo",exLayerCaptionLeft) = "width-twidth"
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
    .Expanded = True
    .MinVisibleCount = 6
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4"

.set_Caption(exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCaption,
 is a caption to be displayed on the control's background.")

.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exL
 is an extra caption to be displayed on the control's background.")

.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exL



.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exL



.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnur
<img>logo:64</img>")
```

```
.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnur



.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnur
twidth")
   .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .BeginUpdate()
   .Expanded = True
   .MinVisibleCount = 6
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
   .set_Caption(EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCaption,"This
is a caption to be displayed on the control's background.")

.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCap
 is an extra caption to be displayed on the control's background.")

.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCap



.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCap



.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayer
<img>logo:64</img>")

.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayer



.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayer
twidth")
   .EndUpdate()
```

```
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->PutMinVisibleCount(6);
spRadialMenu1->GetItems()->PutToString(L"Item 1,Item 2,Item 3,Item 4");
spRadialMenu1->PutCaption(EXRADIALMENULib::exLayerCaption,"This is a caption
to be displayed on the control's background.");
spRadialMenu1->PutExtraCaption("extra",EXRADIALMENULib::exLayerCaption,"This
is an extra caption to be displayed on the control's background.");
spRadialMenu1-
>PutExtraCaption("extra",EXRADIALMENULib::exLayerCaptionAnchor,long(2));
spRadialMenu1-
>PutExtraCaption("extra",EXRADIALMENULib::exLayerCaptionWordWrap,VARIANT_TI

spRadialMenu1-
>PutExtraCaption("extralogo",EXRADIALMENULib::exLayerCaption,"
<img>logo:64</img>");
spRadialMenu1-
>PutExtraCaption("extralogo",EXRADIALMENULib::exLayerCaptionAnchor,long(4));
spRadialMenu1-
>PutExtraCaption("extralogo",EXRADIALMENULib::exLayerCaptionLeft,"width-
twidth");
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
RadialMenu1->MinVisibleCount = 6;
RadialMenu1->Items->ToString = L"Item 1,Item 2,Item 3,Item 4";
RadialMenu1-
>Caption[Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLayerCaption] =
TVariant("This is a caption to be displayed on the control's background.");
RadialMenu1-
>ExtraCaption[TVariant("extra"),Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLa
 = TVariant("This is an extra caption to be displayed on the control's background.");
RadialMenu1-
>ExtraCaption[TVariant("extra"),Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLa
 = TVariant(2);
RadialMenu1-
>ExtraCaption[TVariant("extra"),Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLa
 = TVariant(true);
RadialMenu1-
>ExtraCaption[TVariant("extralogo"),Exradialmenulib_tlb::PropertyLayerCaptionEnum:
 = TVariant("<img>logo:64</img>");
RadialMenu1-
>ExtraCaption[TVariant("extralogo"),Exradialmenulib_tlb::PropertyLayerCaptionEnum:
 = TVariant(4);
RadialMenu1-
>ExtraCaption[TVariant("extralogo"),Exradialmenulib_tlb::PropertyLayerCaptionEnum:
 = TVariant("width-twidth");
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exradialmenu1.MinVisibleCount = 6;
exradialmenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4";
exradialmenu1.set_Caption(exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.e
 is a caption to be displayed on the control's background.");
```

```
exradialmenu1.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCa
 is an extra caption to be displayed on the control's background.");
exradialmenu1.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCa

exradialmenu1.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCa

exradialmenu1.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLay
<img>logo:64</img>");
exradialmenu1.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLay

exradialmenu1.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLay
twidth");
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.Expanded = true;
   RadialMenu1.MinVisibleCount = 6;
   RadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4";
   RadialMenu1.Caption(0) = "This is a caption to be displayed on the control's
background.";
   RadialMenu1.ExtraCaption("extra",0) = "This is an extra caption to be displayed
on the control's background.";
   RadialMenu1.ExtraCaption("extra",3) = 2;
   RadialMenu1.ExtraCaption("extra",8) = true;
   RadialMenu1.ExtraCaption("extralogo",0) = "<img>logo:64</img>";
   RadialMenu1.ExtraCaption("extralogo",3) = 4;
   RadialMenu1.ExtraCaption("extralogo",4) = "width-twidth";
```

```
    RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With RadialMenu1
        .BeginUpdate
        .Expanded = True
        .MinVisibleCount = 6
        .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
        .Caption(0) = "This is a caption to be displayed on the control's background."
        .ExtraCaption("extra",0) = "This is an extra caption to be displayed on the
control's background."
        .ExtraCaption("extra",3) = 2
        .ExtraCaption("extra",8) = True
        .ExtraCaption("extralogo",0) = "<img>logo:64</img>"
        .ExtraCaption("extralogo",3) = 4
        .ExtraCaption("extralogo",4) = "width-twidth"
        .EndUpdate
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
```

```
axRadialMenu1.MinVisibleCount = 6;
axRadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4";
axRadialMenu1.set_Caption(EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC
 is a caption to be displayed on the control's background.");
axRadialMenu1.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEn
 is an extra caption to be displayed on the control's background.");
axRadialMenu1.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEn

axRadialMenu1.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEn

axRadialMenu1.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCapti
<img>logo:64</img>");
axRadialMenu1.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCapti

axRadialMenu1.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCapti
twidth");
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    ;

    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.Expanded(true);
    exradialmenu1.MinVisibleCount(6);
    exradialmenu1.Items().ToString("Item 1,Item 2,Item 3,Item 4");
    exradialmenu1.Caption(0/*exLayerCaption*/,"This is a caption to be displayed on
the control's background.");
    exradialmenu1.ExtraCaption("extra",0/*exLayerCaption*/,"This is an extra caption
to be displayed on the control's background.");

exradialmenu1.ExtraCaption("extra",3/*exLayerCaptionAnchor*/,COMVariant::createFr
```

```
exradialmenu1.ExtraCaption("extra",8/*exLayerCaptionWordWrap*/,COMVariant::creat

   exradialmenu1.ExtraCaption("extralogo",0/*exLayerCaption*/,"
<img>logo:64</img>");

exradialmenu1.ExtraCaption("extralogo",3/*exLayerCaptionAnchor*/,COMVariant::crea

   exradialmenu1.ExtraCaption("extralogo",4/*exLayerCaptionLeft*/,"width-twidth");
   exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   MinVisibleCount := 6;
   Items.ToString := 'Item 1,Item 2,Item 3,Item 4';
   set_Caption(EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCaption,'This
is a caption to be displayed on the control''s background.');

set_ExtraCaption('extra',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCapti
 is an extra caption to be displayed on the control''s background.');

set_ExtraCaption('extra',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCapti



set_ExtraCaption('extra',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCapti



set_ExtraCaption('extralogo',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC



set_ExtraCaption('extralogo',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC
```

```
set_ExtraCaption('extralogo',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC
twidth');
   EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   MinVisibleCount := 6;
   Items.ToString := 'Item 1,Item 2,Item 3,Item 4';
   Caption[EXRADIALMENULib_TLB.exLayerCaption] := 'This is a caption to be
displayed on the control''s background.';
   ExtraCaption['extra',EXRADIALMENULib_TLB.exLayerCaption] := 'This is an extra
caption to be displayed on the control''s background.';
   ExtraCaption['extra',EXRADIALMENULib_TLB.exLayerCaptionAnchor] :=
OleVariant(2);
   ExtraCaption['extra',EXRADIALMENULib_TLB.exLayerCaptionWordWrap] :=
OleVariant(True);
   ExtraCaption['extralogo',EXRADIALMENULib_TLB.exLayerCaption] :=
'<img>logo:64</img>';
   ExtraCaption['extralogo',EXRADIALMENULib_TLB.exLayerCaptionAnchor] :=
OleVariant(4);
   ExtraCaption['extralogo',EXRADIALMENULib_TLB.exLayerCaptionLeft] := 'width-
twidth';
   EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
   .BeginUpdate
   .Expanded = .T.
   .MinVisibleCount = 6
```

```
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
   .Object.Caption(0) = "This is a caption to be displayed on the control's
background."
   .Object.ExtraCaption("extra",0) = "This is an extra caption to be displayed on the
control's background."
   .Object.ExtraCaption("extra",3) = 2
   .Object.ExtraCaption("extra",8) = .T.
   .Object.ExtraCaption("extralogo",0) = "<img>logo:64</img>"
   .Object.ExtraCaption("extralogo",3) = 4
   .Object.ExtraCaption("extralogo",4) = "width-twidth"
   .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.MinVisibleCount = 6
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4"
oRadialMenu.Template = [Caption(0) = "This is a caption to be displayed on the
control's background."] // oRadialMenu.Caption(0) = "This is a caption to be
displayed on the control's background."
oRadialMenu.Template = [ExtraCaption("extra",0) = "This is an extra caption to be
displayed on the control's background."] // oRadialMenu.ExtraCaption("extra",0) =
"This is an extra caption to be displayed on the control's background."
oRadialMenu.Template = [ExtraCaption("extra",3) = 2] //
oRadialMenu.ExtraCaption("extra",3) = 2
oRadialMenu.Template = [ExtraCaption("extra",8) = True] //
oRadialMenu.ExtraCaption("extra",8) = true
oRadialMenu.Template = [ExtraCaption("extralogo",0) = "<img>logo:64</img>"] //
oRadialMenu.ExtraCaption("extralogo",0) = "<img>logo:64</img>"
oRadialMenu.Template = [ExtraCaption("extralogo",3) = 4] //
oRadialMenu.ExtraCaption("extralogo",3) = 4
oRadialMenu.Template = [ExtraCaption("extralogo",4) = "width-twidth"] //
```

```
oRadialMenu.ExtraCaption("extralogo",4) = "width-twidth"
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = .t.
oRadialMenu.MinVisibleCount = 6
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4"
oRadialMenu.Template = "Caption(0) = `This is a caption to be displayed on the control's background.`" // oRadialMenu.Caption(0) = "This is a caption to be displayed on the control's background."
oRadialMenu.Template = "ExtraCaption(`extra`,0) = `This is an extra caption to be displayed on the control's background.`" // oRadialMenu.ExtraCaption("extra",0) = "This is an extra caption to be displayed on the control's background."
oRadialMenu.Template = "ExtraCaption(`extra`,3) = 2" // oRadialMenu.ExtraCaption("extra",3) = 2
oRadialMenu.Template = "ExtraCaption(`extra`,8) = True" // oRadialMenu.ExtraCaption("extra",8) = .t.
oRadialMenu.Template = "ExtraCaption(`extralogo`,0) = `<img>logo:64</img>`" // oRadialMenu.ExtraCaption("extralogo",0) = "<img>logo:64</img>"
oRadialMenu.Template = "ExtraCaption(`extralogo`,3) = 4" // oRadialMenu.ExtraCaption("extralogo",3) = 4
oRadialMenu.Template = "ExtraCaption(`extralogo`,4) = `width-twidth`" // oRadialMenu.ExtraCaption("extralogo",4) = "width-twidth"
oRadialMenu.EndUpdate()
```

Visual Objects

```

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:MinVisibleCount := 6
```

```
oDCOCX_Exontrol1:Items:ToString := "Item 1,Item 2,Item 3,Item 4"
oDCOCX_Exontrol1:[Caption,exLayerCaption] := "This is a caption to be displayed on the control's background."
oDCOCX_Exontrol1:[ExtraCaption,"extra",exLayerCaption] := "This is an extra caption to be displayed on the control's background."
oDCOCX_Exontrol1:[ExtraCaption,"extra",exLayerCaptionAnchor] := 2
oDCOCX_Exontrol1:[ExtraCaption,"extra",exLayerCaptionWordWrap] := true
oDCOCX_Exontrol1:[ExtraCaption,"extralogo",exLayerCaption] := "<img>logo:64</img>"
oDCOCX_Exontrol1:[ExtraCaption,"extralogo",exLayerCaptionAnchor] := 4
oDCOCX_Exontrol1:[ExtraCaption,"extralogo",exLayerCaptionLeft] := "width-twidth"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.MinVisibleCount = 6
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4"
oRadialMenu.Caption(0,"This is a caption to be displayed on the control's background.")
oRadialMenu.ExtraCaption("extra",0,"This is an extra caption to be displayed on the control's background.")
oRadialMenu.ExtraCaption("extra",3,2)
oRadialMenu.ExtraCaption("extra",8,true)
oRadialMenu.ExtraCaption("extralogo",0,"<img>logo:64</img>")
oRadialMenu.ExtraCaption("extralogo",3,4)
oRadialMenu.ExtraCaption("extralogo",4,"width-twidth")
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
```

```
    Forward Send OnCreate
    Send ComBeginUpdate
    Set ComExpanded to True
    Set ComMinVisibleCount to 6
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Set ComToString of hoItems to "Item 1,Item 2,Item 3,Item 4"
    Send Destroy to hoItems
    Set ComCaption OLEexLayerCaption to "This is a caption to be displayed on the
control's background."
    Set ComExtraCaption "extra" OLEexLayerCaption to "This is an extra caption to be
displayed on the control's background."
    Set ComExtraCaption "extra" OLEexLayerCaptionAnchor to 2
    Set ComExtraCaption "extra" OLEexLayerCaptionWordWrap to True
    Set ComExtraCaption "extralogo" OLEexLayerCaption to "<img>logo:64</img>"
    Set ComExtraCaption "extralogo" OLEexLayerCaptionAnchor to 4
    Set ComExtraCaption "extralogo" OLEexLayerCaptionLeft to "width-twidth"
    Send ComEndUpdate
End_Procedure
```

XBase++

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

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

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

```
   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
      oRadialMenu:MinVisibleCount := 6
      oRadialMenu:Items():ToString := "Item 1,Item 2,Item 3,Item 4"
      oRadialMenu:SetProperty("Caption",0/*exLayerCaption*/,"This is a caption to be displayed on the control's background.")
      oRadialMenu:SetProperty("ExtraCaption","extra",0/*exLayerCaption*/,"This is an extra caption to be displayed on the control's background.")
      oRadialMenu:SetProperty("ExtraCaption","extra",3/*exLayerCaptionAnchor*/,2)

oRadialMenu:SetProperty("ExtraCaption","extra",8/*exLayerCaptionWordWrap*/,.T.)
      oRadialMenu:SetProperty("ExtraCaption","extralogo",0/*exLayerCaption*/,"<img>logo:64</img>")

oRadialMenu:SetProperty("ExtraCaption","extralogo",3/*exLayerCaptionAnchor*/,4)

oRadialMenu:SetProperty("ExtraCaption","extralogo",4/*exLayerCaptionLeft*/,"width-twidth")
      oRadialMenu:EndUpdate()

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

property RadialMenu.CustomBackAlpha as Byte

Specifies the value of alpha / opacity channel to show the custom portion of the radial menu.

Type	Description
Byte	A BYTE expression that specifies the value of alpha / opacity channel to show the custom portion of the radial menu. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

By default, The CustomBackAlpha property is 255. The CustomBackColor / CustomBackAlpha property specifies the color to show the custom portion of the radial menu. The CustomPicture property indicates the picture to be shown on the custom's background. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

The ItemsPicture property indicates the picture to be shown on the items's background. The BackgroundPicture property indicates the picture to be shown on the radial menu's background. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu.

property RadialMenu.CustomBackColor as Color

Specifies the color to show the custom portion of the radial menu.

Type	Description
Color	A Color expression that specifies the color to show the custom portion of the radial menu. If -1, no solid color is applied on the custom portion of the control.

By default, The CustomBackColor property is -1, so no background color is applied on the custom section of the control. The CustomBackColor / CustomBackAlpha property Specifies the color to show the custom portion of the radial menu. The CustomPicture property indicates the picture to be shown on the custom's background. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

The ItemsPicture property indicates the picture to be shown on the items's background. The BackgroundPicture property indicates the picture to be shown on the radial menu's background. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu.

property RadialMenu.CustomHeight as Long

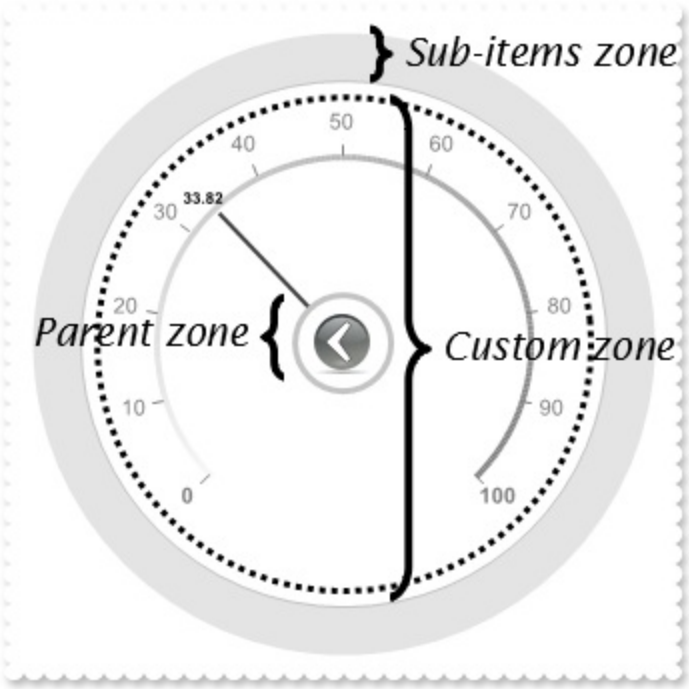
Gets a value that represents the height of the inner custom control.

Type	Description
Long	A Long expression that specifies the height of the custom-section of the control.

The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The following properties determine the position / size of the custom section of the control:

- CustomLeft property, represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomTop property, represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomWidth property, represents the width of the inner custom control
- CustomHeight property, represents the height of the inner custom control

The following screen show shows the custom-section of the control:



property RadialMenu.CustomLeft as Long

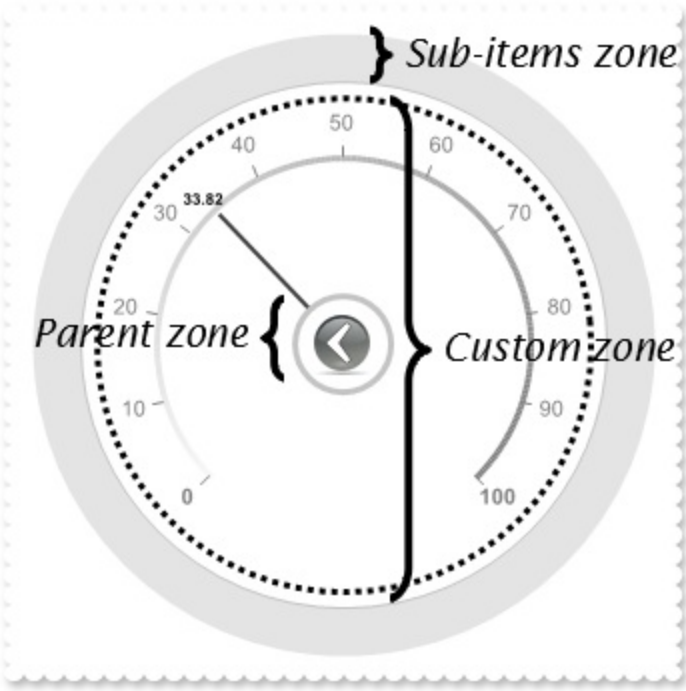
Gets a value that represents the distance between the left side of the inner custom control and the left side of the control itself.

Type	Description
Long	A Long expression that represents the distance between the left side of the inner custom control and the left side of the control itself.

The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The following properties determine the position / size of the custom section of the control:

- CustomLeft property, represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomTop property, represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomWidth property, represents the width of the inner custom control
- CustomHeight property, represents the height of the inner custom control

The following screen show shows the custom-section of the control:



property RadialMenu.CustomPicture as Variant

Indicates the picture to be shown on the custom's background.

Type	Description

Variant	A VARIANT expression that specifies the icon/picture/image to be displayed as described:
	
	• A String expression indicates:
	  ◦ a name of a picture file in the PicturePath folder. For instance, CustomPicture = "favorites.png", loads the favorites.png file if found in the PicturePath folder.
	  ◦ a picture file including its absolute path. For instance, CustomPicture = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor loads the favorites.png file from absolute path
	  ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, CustomPicture = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor
	  ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on.
	• A Picture object that indicates the picture to be displayed. For instance, CustomPicture = LoadPicture("picture.jpg")
	
	If no picture/image is found, the items section displays no picture/image.

By default, The CustomPicture property is empty. The CustomPicture property indicates the picture to be shown on the custom's background. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The CustomBackColor / CustomBackAlpha property Specifies the color to show the custom portion of the radial menu.

The ItemsPicture property indicates the picture to be shown on the items's background. The BackgroundPicture property indicates the picture to be shown on the radial menu's background. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu.

The following screen shot shows control with no background picture (default):

The following screen shot shows control with a background picture, for the custom section:

property RadialMenu.CustomTop as Long

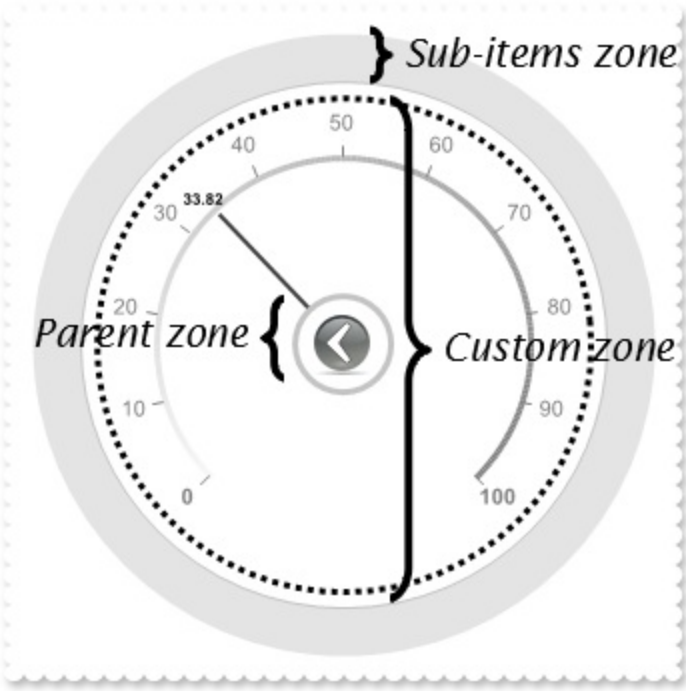
Gets a value that represents the distance between the left side of the inner custom control and the left side of the control itself.

Type	Description
Long	A Long expression that represents the distance between the left side of the inner custom control and the left side of the control itself.

The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The following properties determine the position / size of the custom section of the control:

- CustomLeft property, represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomTop property, represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomWidth property, represents the width of the inner custom control
- CustomHeight property, represents the height of the inner custom control

The following screen show shows the custom-section of the control:



property RadialMenu.CustomWidth as Long

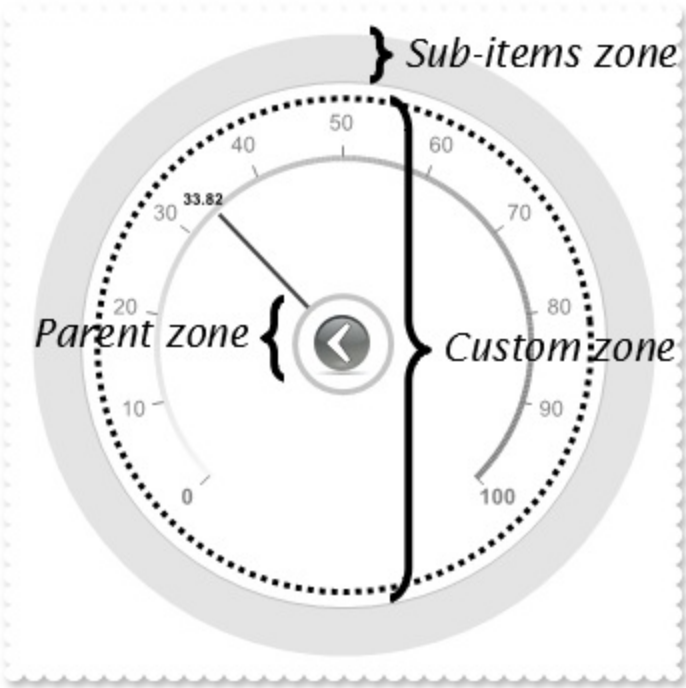
Gets a value that represents the width of the inner custom control.

Type	Description
Long	A Long expression that represents the width of the inner custom control.

The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The following properties determine the position / size of the custom section of the control:

- CustomLeft property, gets a value that represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomTop property, gets a value that represents the distance between the left side of the inner custom control and the left side of the control itself
- CustomWidth property, gets a value that represents the width of the inner custom control
- CustomHeight property, gets a value that represents the height of the inner custom control

The following screen show shows the custom-section of the control:



property RadialMenu.DisplayAngle as Double

Specifies the angle to display the items around the radial menu.

Type	Description
Double	A double expression that specifies the angle to display the items around the radial menu.

By default, the DisplayAngle property is 0 degree. The DisplayAngle property specifies the angle to display the items around the radial menu. The DisplayRadial property determines how the item is displayed on the radial menu.

The following screen show shows the control using the DisplayAngle property on 0 degree (by default):

The following screen show shows the control using the DisplayAngle property on -10 degree:

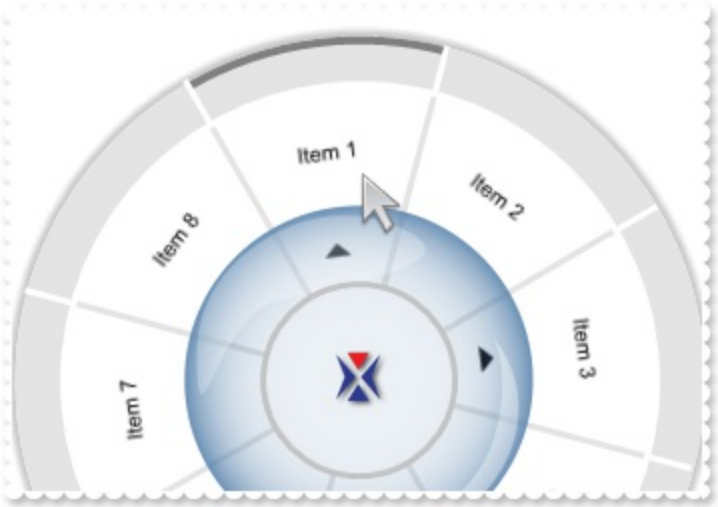

Foreground-Color
Background-Color
Tag
Font
List
Undo
Copy
Redo

property RadialMenu.DisplayArrow as RadialItemsEnum

Indicates where the arrow of items with children is displayed.

Type	Description
RadialItemsEnum	A RadialItemsEnum expression that indicates where the arrow of items with children is displayed.

By default, the DisplayArrow property is exRadialSubItems, which specifies that the ""arrow" for child-items is displayed in the sub-items zone of the control. The DisplayArrow property indicates whether the "arrow" HTML picture is displayed on the items/sub-items zone of the control, or in both. The DisplayCenterArrow property specifies the ratio to determine where the arrow of items with children is displayed.



The following samples show how you can display the "arrow" into the items section of the control:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   .DisplayAngle = -7.5
   .DisplayArrow = 1
   .DisplayRadial(1) = 1
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .ParentSize = "48 * dpi"
   .ParentPicture = "Background\frontb.png"
   .ArrowImage = "arrow.png"
   .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
```

```
    .InflateParentPicture = "72 * dpi"
    .EndUpdate
End With
```

VB6

```
With RadialMenu1
    .BeginUpdate
    .Expanded = True
    .DisplayAngle = -7.5
    .DisplayArrow = exRadialItems
    .DisplayRadial(exRadialItems) = exDisplayRadialRotated
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
    .Expanded = True
    .DisplayAngle = -7.5
    .DisplayArrow = exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems

.set_DisplayRadial(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,exontro

    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate()
```

```
End With
```

VB.NET for /COM

```
With AxRadialMenu1
    .BeginUpdate()
    .Expanded = True
    .DisplayAngle = -7.5
    .DisplayArrow = EXRADIALMENULib.RadialItemsEnum.exRadialItems

.set_DisplayRadial(EXRADIALMENULib.RadialItemsEnum.exRadialItems,EXRADIALMENU

    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate()
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->PutDisplayAngle(-7.5);
spRadialMenu1->PutDisplayArrow(EXRADIALMENULib::exRadialItems);
spRadialMenu1-
```

```
>PutDisplayRadial(EXRADIALMENULib::exRadialItems,EXRADIALMENULib::exDisplayRa

spRadialMenu1->PutPicturesPath(L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
spRadialMenu1->PutParentSize(L"48 * dpi");
spRadialMenu1->PutParentPicture("Background\\frontb.png");
spRadialMenu1->PutArrowImage("arrow.png");
spRadialMenu1->GetItems()->PutToString(L"Item 1(1),Item 2,Item 3(1),Item 4,Item
5,Item 6,Item 7,Item 8");
spRadialMenu1->PutInflateParentPicture(L"72 * dpi");
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
RadialMenu1->DisplayAngle = -7.5;
RadialMenu1->DisplayArrow =
Exradialmenulib_tlb::RadialItemsEnum::exRadialItems;
RadialMenu1->DisplayRadial[Exradialmenulib_tlb::RadialItemsEnum::exRadialItems] =
Exradialmenulib_tlb::DisplayRadialEnum::exDisplayRadialRotated;
RadialMenu1->PicturesPath = L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
RadialMenu1->ParentSize = L"48 * dpi";
RadialMenu1->set_ParentPicture(TVariant("Background\\frontb.png"));
RadialMenu1->set_ArrowImage(TVariant("arrow.png"));
RadialMenu1->Items->ToString = L"Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8";
RadialMenu1->InflateParentPicture = L"72 * dpi";
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exradialmenu1.DisplayAngle = -7.5;
```

```
exradialmenu1.DisplayArrow =
exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems;
exradialmenu1.set_DisplayRadial(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
exradialmenu1.ParentSize = "48 * dpi";
exradialmenu1.ParentPicture = "Background\\frontb.png";
exradialmenu1.ArrowImage = "arrow.png";
exradialmenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
exradialmenu1.InflateParentPicture = "72 * dpi";
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.Expanded = true;
   RadialMenu1.DisplayAngle = -7.5;
   RadialMenu1.DisplayArrow = 1;
   RadialMenu1.DisplayRadial(1) = 1;
   RadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
   RadialMenu1.ParentSize = "48 * dpi";
   RadialMenu1.ParentPicture = "Background\\frontb.png";
   RadialMenu1.ArrowImage = "arrow.png";
   RadialMenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
   RadialMenu1.InflateParentPicture = "72 * dpi";
```

```
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .Expanded = True
      .DisplayAngle = -7.5
      .DisplayArrow = 1
      .DisplayRadial(1) = 1
      .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
      .ParentSize = "48 * dpi"
      .ParentPicture = "Background\frontb.png"
      .ArrowImage = "arrow.png"
      .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
      .InflateParentPicture = "72 * dpi"
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
axRadialMenu1.DisplayAngle = -7.5;
```

```
axRadialMenu1.DisplayArrow = EXRADIALMENULib.RadialItemsEnum.exRadialItems;
axRadialMenu1.set_DisplayRadial(EXRADIALMENULib.RadialItemsEnum.exRadialItems,
axRadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
axRadialMenu1.ParentSize = "48 * dpi";
axRadialMenu1.ParentPicture = "Background\\frontb.png";
axRadialMenu1.ArrowImage = "arrow.png";
axRadialMenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
axRadialMenu1.InflateParentPicture = "72 * dpi";
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    ;

    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.Expanded(true);
    exradialmenu1.DisplayAngle(-7.5);
    exradialmenu1.DisplayArrow(1/*exRadialItems*/);
    exradialmenu1.DisplayRadial(1/*exRadialItems*/,1/*exDisplayRadialRotated*/);
    exradialmenu1.PicturesPath("C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
    exradialmenu1.ParentSize("48 * dpi");
    exradialmenu1.ParentPicture("Background\\frontb.png");
    exradialmenu1.ArrowImage("arrow.png");
    exradialmenu1.Items().ToString("Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8");
    exradialmenu1.InflateParentPicture("72 * dpi");
    exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   DisplayAngle := -7.5;
   DisplayArrow := EXRADIALMENULib.RadialItemsEnum.exRadialItems;

set_DisplayRadial(EXRADIALMENULib.RadialItemsEnum.exRadialItems,EXRADIALMENU

   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
   ParentSize := '48 * dpi';
   ParentPicture := 'Background\frontb.png';
   ArrowImage := 'arrow.png';
   Items.ToString := 'Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8';
   InflateParentPicture := '72 * dpi';
   EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   DisplayAngle := -7.5;
   DisplayArrow := EXRADIALMENULib_TLB.exRadialItems;
   DisplayRadial[EXRADIALMENULib_TLB.exRadialItems] :=
EXRADIALMENULib_TLB.exDisplayRadialRotated;
   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
   ParentSize := '48 * dpi';
   ParentPicture := 'Background\frontb.png';
   ArrowImage := 'arrow.png';
   Items.ToString := 'Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8';
   InflateParentPicture := '72 * dpi';
   EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
   .BeginUpdate
   .Expanded = .T.
   .DisplayAngle = -7.5
   .DisplayArrow = 1
   .Object.DisplayRadial(1) = 1
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .ParentSize = "48 * dpi"
   .ParentPicture = "Background\frontb.png"
   .ArrowImage = "arrow.png"
   .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
   .InflateParentPicture = "72 * dpi"
   .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.DisplayAngle = -7.5
oRadialMenu.DisplayArrow = 1
oRadialMenu.Template = [DisplayRadial(1) = 1] // oRadialMenu.DisplayRadial(1) = 1
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.ParentSize = "48 * dpi"
oRadialMenu.ParentPicture = "Background\frontb.png"
oRadialMenu.ArrowImage = "arrow.png"
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8"
oRadialMenu.InflateParentPicture = "72 * dpi"
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = .t.
oRadialMenu.DisplayAngle = -7.5
oRadialMenu.DisplayArrow = 1
oRadialMenu.Template = "DisplayRadial(1) = 1" // oRadialMenu.DisplayRadial(1) = 1
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.ParentSize = "48 * dpi"
oRadialMenu.ParentPicture = "Background\frontb.png"
oRadialMenu.ArrowImage = "arrow.png"
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8"
oRadialMenu.InflateParentPicture = "72 * dpi"
oRadialMenu.EndUpdate()
```

Visual Objects

```

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:DisplayAngle := -7.5
oDCOCX_Exontrol1:DisplayArrow := exRadialItems
oDCOCX_Exontrol1:[DisplayRadial,exRadialItems] := exDisplayRadialRotated
oDCOCX_Exontrol1:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oDCOCX_Exontrol1:ParentSize := "48 * dpi"
oDCOCX_Exontrol1:ParentPicture := "Background\frontb.png"
oDCOCX_Exontrol1:ArrowImage := "arrow.png"
oDCOCX_Exontrol1:Items:ToString := "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8"
oDCOCX_Exontrol1:InflateParentPicture := "72 * dpi"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.DisplayAngle = -7.5
oRadialMenu.DisplayArrow = 1
oRadialMenu.DisplayRadial(1,1)
oRadialMenu.PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.ParentSize = "48 * dpi"
oRadialMenu.ParentPicture = "Background\frontb.png"
oRadialMenu.ArrowImage = "arrow.png"
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
oRadialMenu.InflateParentPicture = "72 * dpi"
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComExpanded to True
   Set ComDisplayAngle to -7.5
   Set ComDisplayArrow to OLEexRadialItems
   Set ComDisplayRadial OLEexRadialItems to OLEexDisplayRadialRotated
   Set ComPicturesPath to "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   Set ComParentSize to "48 * dpi"
   Set ComParentPicture to "Background\frontb.png"
   Set ComArrowImage to "arrow.png"
   Variant voItems
   Get ComItems to voItems
```

```
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Set ComToString of hoItems to "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8"
    Send Destroy to hoItems
    Set ComInflateParentPicture to "72 * dpi"
    Send ComEndUpdate
End_Procedure
```

XBase++

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

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

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
      oRadialMenu:DisplayAngle := -7.5
      oRadialMenu:DisplayArrow := 1/*exRadialItems*/

oRadialMenu:SetProperty("DisplayRadial",1/*exRadialItems*/,1/*exDisplayRadialRotate
```

```
      oRadialMenu:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
      oRadialMenu:ParentSize := "48 * dpi"
      oRadialMenu:ParentPicture := "Background\frontb.png"
      oRadialMenu:ArrowImage := "arrow.png"
      oRadialMenu:Items():ToString := "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8"
      oRadialMenu:InflateParentPicture := "72 * dpi"
      oRadialMenu:EndUpdate()

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

property RadialMenu.DisplayCenter(Type as RadialItemsEnum) as Double

Specifies the ratio to determine where the image/caption of the item is displayed.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that indicates the part of the item to be changed.
Double	A Double expression that specifies the ratio to determine where the image/caption of the item is displayed.

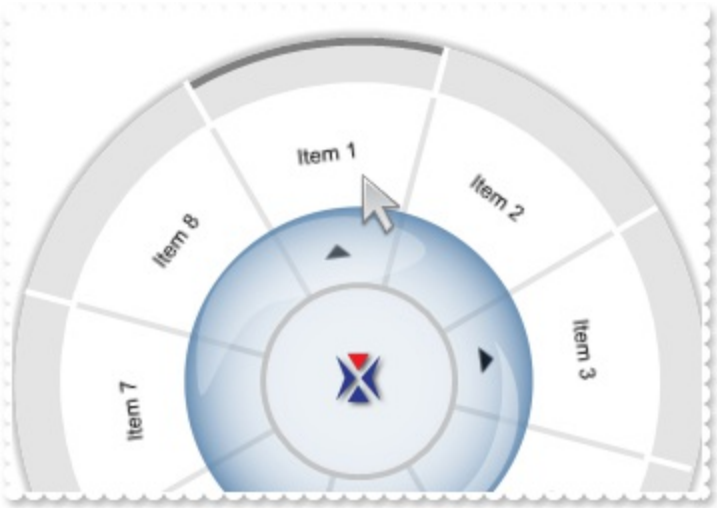
By default, the DisplayCenter(exRadialSubItems) property is 0.5, while the the DisplayCenter(exRadialItems) property is 0.35. The DisplayCenter property specifies the ratio to determine where the image/caption of the item is displayed. For instance, you can use the DisplayCenter property to display the item closer to parent or sub-items/border section of the control. The Caption property retrieves or sets a value that indicates the item's caption. You can specify the caption of the item using the Caption parameter of the Add method. The ForeColor property specifies the item's foreground color. The Image property assigns an icon/picture to the item. The Name property of the Item object is equivalent with the Caption(exRadialItems) property. The UserData property retrieves or sets a value that indicates the item's user data.

property RadialMenu.DisplayCenterArrow(Type as RadialItemsEnum) as Double

Specifies the ratio to determine where the arrow of items with children is displayed.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies the zone of the control to be changed.
Double	A Double expression that specifies the ratio to determine where the arrow of items with children is displayed.

By default, the DisplayCenterArrow(exRadialSubItems) property is 0.5, while the the DisplayCenterArrow(exRadialItems) property is 1. The DisplayCenterArrow property specifies the ratio to determine where the arrow of items with children is displayed. The DisplayArrow property indicates whether the "arrow" HTML picture is displayed on the items/sub-items zone of the control, or in both. The DisplayCenter property specifies the ratio to determine where the image/caption of the item is displayed.



The following samples show how you can display the "arrow" into the items section of the control:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   .DisplayAngle = -7.5
   .DisplayArrow = 1
   .DisplayRadial(1) = 1
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
```

```
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate
End With
```

VB6

```
With RadialMenu1
    .BeginUpdate
    .Expanded = True
    .DisplayAngle = -7.5
    .DisplayArrow = exRadialItems
    .DisplayRadial(exRadialItems) = exDisplayRadialRotated
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
    .Expanded = True
    .DisplayAngle = -7.5
    .DisplayArrow = exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems

.set_DisplayRadial(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,exontro

    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
```

```
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
    .BeginUpdate()
    .Expanded = True
    .DisplayAngle = -7.5
    .DisplayArrow = EXRADIALMENULib.RadialItemsEnum.exRadialItems

.set_DisplayRadial(EXRADIALMENULib.RadialItemsEnum.exRadialItems,EXRADIALMENU

    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate()
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
```

```
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->PutDisplayAngle(-7.5);
spRadialMenu1->PutDisplayArrow(EXRADIALMENULib::exRadialItems);
spRadialMenu1-
>PutDisplayRadial(EXRADIALMENULib::exRadialItems,EXRADIALMENULib::exDisplayRa

spRadialMenu1->PutPicturesPath(L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
spRadialMenu1->PutParentSize(L"48 * dpi");
spRadialMenu1->PutParentPicture("Background\\frontb.png");
spRadialMenu1->PutArrowImage("arrow.png");
spRadialMenu1->GetItems()->PutToString(L"Item 1(1),Item 2,Item 3(1),Item 4,Item
5,Item 6,Item 7,Item 8");
spRadialMenu1->PutInflateParentPicture(L"72 * dpi");
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
RadialMenu1->DisplayAngle = -7.5;
RadialMenu1->DisplayArrow =
ExradialmenulibTlb::RadialItemsEnum::exRadialItems;
RadialMenu1->DisplayRadial[ExradialmenulibTlb::RadialItemsEnum::exRadialItems] =
ExradialmenulibTlb::DisplayRadialEnum::exDisplayRadialRotated;
RadialMenu1->PicturesPath = L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
RadialMenu1->ParentSize = L"48 * dpi";
RadialMenu1->set_ParentPicture(TVariant("Background\\frontb.png"));
RadialMenu1->set_ArrowImage(TVariant("arrow.png"));
RadialMenu1->Items->ToString = L"Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8";
RadialMenu1->InflateParentPicture = L"72 * dpi";
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exradialmenu1.DisplayAngle = -7.5;
exradialmenu1.DisplayArrow =
exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems;
exradialmenu1.set_DisplayRadial(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
exradialmenu1.ParentSize = "48 * dpi";
exradialmenu1.ParentPicture = "Background\\frontb.png";
exradialmenu1.ArrowImage = "arrow.png";
exradialmenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
exradialmenu1.InflateParentPicture = "72 * dpi";
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
    RadialMenu1.BeginUpdate();
    RadialMenu1.Expanded = true;
    RadialMenu1.DisplayAngle = -7.5;
    RadialMenu1.DisplayArrow = 1;
    RadialMenu1.DisplayRadial(1) = 1;
    RadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
    RadialMenu1.ParentSize = "48 * dpi";
    RadialMenu1.ParentPicture = "Background\\frontb.png";
    RadialMenu1.ArrowImage = "arrow.png";
```

```
   RadialMenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8";
   RadialMenu1.InflateParentPicture = "72 * dpi";
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C" id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .Expanded = True
      .DisplayAngle = -7.5
      .DisplayArrow = 1
      .DisplayRadial(1) = 1
      .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
      .ParentSize = "48 * dpi"
      .ParentPicture = "Background\frontb.png"
      .ArrowImage = "arrow.png"
      .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
      .InflateParentPicture = "72 * dpi"
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
axRadialMenu1.DisplayAngle = -7.5;
axRadialMenu1.DisplayArrow = EXRADIALMENULib.RadialItemsEnum.exRadialItems;
axRadialMenu1.set_DisplayRadial(EXRADIALMENULib.RadialItemsEnum.exRadialItems,

axRadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
axRadialMenu1.ParentSize = "48 * dpi";
axRadialMenu1.ParentPicture = "Background\\frontb.png";
axRadialMenu1.ArrowImage = "arrow.png";
axRadialMenu1.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8";
axRadialMenu1.InflateParentPicture = "72 * dpi";
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    ;

    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.Expanded(true);
    exradialmenu1.DisplayAngle(-7.5);
    exradialmenu1.DisplayArrow(1/*exRadialItems*/);
    exradialmenu1.DisplayRadial(1/*exRadialItems*/,1/*exDisplayRadialRotated*/);
    exradialmenu1.PicturesPath("C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
    exradialmenu1.ParentSize("48 * dpi");
    exradialmenu1.ParentPicture("Background\\frontb.png");
    exradialmenu1.ArrowImage("arrow.png");
    exradialmenu1.Items().ToString("Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8");
```

```
    exradialmenu1.InflateParentPicture("72 * dpi");
    exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
    BeginUpdate();
    Expanded := True;
    DisplayAngle := -7.5;
    DisplayArrow := EXRADIALMENULib.RadialItemsEnum.exRadialItems;

set_DisplayRadial(EXRADIALMENULib.RadialItemsEnum.exRadialItems,EXRADIALMENU

    PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
    ParentSize := '48 * dpi';
    ParentPicture := 'Background\frontb.png';
    ArrowImage := 'arrow.png';
    Items.ToString := 'Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8';
    InflateParentPicture := '72 * dpi';
    EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
    BeginUpdate();
    Expanded := True;
    DisplayAngle := -7.5;
    DisplayArrow := EXRADIALMENULib_TLB.exRadialItems;
    DisplayRadial[EXRADIALMENULib_TLB.exRadialItems] :=
EXRADIALMENULib_TLB.exDisplayRadialRotated;
    PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
    ParentSize := '48 * dpi';
    ParentPicture := 'Background\frontb.png';
    ArrowImage := 'arrow.png';
```

```
    Items.ToString := 'Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8';
    InflateParentPicture := '72 * dpi';
    EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
    .BeginUpdate
    .Expanded = .T.
    .DisplayAngle = -7.5
    .DisplayArrow = 1
    .Object.DisplayRadial(1) = 1
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .ParentSize = "48 * dpi"
    .ParentPicture = "Background\frontb.png"
    .ArrowImage = "arrow.png"
    .Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
    .InflateParentPicture = "72 * dpi"
    .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.DisplayAngle = -7.5
oRadialMenu.DisplayArrow = 1
oRadialMenu.Template = [DisplayRadial(1) = 1] // oRadialMenu.DisplayRadial(1) = 1
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.ParentSize = "48 * dpi"
oRadialMenu.ParentPicture = "Background\frontb.png"
oRadialMenu.ArrowImage = "arrow.png"
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
```

```
7,Item 8"
oRadialMenu.InflateParentPicture = "72 * dpi"
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = .t.
oRadialMenu.DisplayAngle = -7.5
oRadialMenu.DisplayArrow = 1
oRadialMenu.Template = "DisplayRadial(1) = 1" // oRadialMenu.DisplayRadial(1) = 1
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.ParentSize = "48 * dpi"
oRadialMenu.ParentPicture = "Background\frontb.png"
oRadialMenu.ArrowImage = "arrow.png"
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item
7,Item 8"
oRadialMenu.InflateParentPicture = "72 * dpi"
oRadialMenu.EndUpdate()
```

Visual Objects

```

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:DisplayAngle := -7.5
oDCOCX_Exontrol1:DisplayArrow := exRadialItems
oDCOCX_Exontrol1:[DisplayRadial,exRadialItems] := exDisplayRadialRotated
oDCOCX_Exontrol1:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oDCOCX_Exontrol1:ParentSize := "48 * dpi"
oDCOCX_Exontrol1:ParentPicture := "Background\frontb.png"
```

```
oDCOCX_Exontrol1:ArrowImage := "arrow.png"
oDCOCX_Exontrol1:Items:ToString := "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
oDCOCX_Exontrol1:InflateParentPicture := "72 * dpi"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.DisplayAngle = -7.5
oRadialMenu.DisplayArrow = 1
oRadialMenu.DisplayRadial(1,1)
oRadialMenu.PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.ParentSize = "48 * dpi"
oRadialMenu.ParentPicture = "Background\frontb.png"
oRadialMenu.ArrowImage = "arrow.png"
oRadialMenu.Items.ToString = "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item 6,Item 7,Item 8"
oRadialMenu.InflateParentPicture = "72 * dpi"
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComExpanded to True
   Set ComDisplayAngle to -7.5
   Set ComDisplayArrow to OLEexRadialItems
   Set ComDisplayRadial OLEexRadialItems to OLEexDisplayRadialRotated
   Set ComPicturesPath to "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
```

```
    Set ComParentSize to "48 * dpi"
    Set ComParentPicture to "Background\frontb.png"
    Set ComArrowImage to "arrow.png"
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Set ComToString of hoItems to "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8"
    Send Destroy to hoItems
    Set ComInflateParentPicture to "72 * dpi"
    Send ComEndUpdate
End_Procedure
```

XBase++

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

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

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
```

```
      oRadialMenu:DisplayAngle := -7.5
      oRadialMenu:DisplayArrow := 1/*exRadialItems*/

oRadialMenu:SetProperty("DisplayRadial",1/*exRadialItems*/,1/*exDisplayRadialRotate

      oRadialMenu:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
      oRadialMenu:ParentSize := "48 * dpi"
      oRadialMenu:ParentPicture := "Background\frontb.png"
      oRadialMenu:ArrowImage := "arrow.png"
      oRadialMenu:Items():ToString := "Item 1(1),Item 2,Item 3(1),Item 4,Item 5,Item
6,Item 7,Item 8"
      oRadialMenu:InflateParentPicture := "72 * dpi"
      oRadialMenu:EndUpdate()

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

property RadialMenu.DisplayRadial(Type as RadialItemsEnum) as DisplayRadialEnum

Determines how the item is displayed on the radial menu.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies which portion of the item is displayed.
DisplayRadialEnum	A DisplayRadialEnum expression that determines how the item is displayed on the radial menu.

By default, the DisplayRadial(exRadialItems) property is exDisplayRadialFlat, while the DisplayRadial(exRadialSubItems) property is exDisplayRadialRotated270. The DisplayRadial property determines how the item is displayed on the radial menu. The DisplayAngle property specifies the angle to display the items around the radial menu.

The following screen shot shows items, while DisplayRadial(exRadialItems) property is exDisplayRadialRotated270.

property RadialMenu.Enabled as Boolean

Enables or disables the control.

Type	Description
Boolean	A boolean expression that determines whether an control can respond to user-generated events.

By default, the Enabled property is True. The Enabled property specifies whether the entire control is enabled or disabled.

method RadialMenu.EndUpdate ()

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

Type	Description

property RadialMenu.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.

property RadialMenu.ExcludeParentFromItems as Boolean

Gets or sets a value that specifies whether the parent portion of the control is excluded from the items zone.

Type	Description
Boolean	A Boolean expression that specifies whether the parent portion of the control is excluded from the items zone.

By default, the ExcludeParentFromItems property is False. The ExcludeParentFromItems property gets or sets a value that specifies whether the parent portion of the control is excluded from the items zone. The The ExcludeParentFromItems property has effect, only if the ParentBackColor property is -1. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The RadialLineColor(exRadialParentBorder) property specifies the color to show the border around the parent section of the control. The RadialLineColor(exRadialHotParent) property specifies the color to show the border around the parent section of the control, while the cursor hovers the parent section.

The following screen shot marks the parent zone (in red), being completed by the items:

method RadialMenu.ExecuteTemplate (Template as String)

Executes a template and returns the result.

Type	Description
Template as String	A Template string being executed

Return	Description
Variant	A Variant expression that indicates the result after executing the Template.

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

For instance, the following sample retrieves the control's background color:

```
Debug.Print RadialMenu1.ExecuteTemplate("BackColor")
```

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

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

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

The Template 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 RadialMenu.Expanded as Boolean

Indicates whether the radial menu is expanded or collapsed.

Type	Description
Boolean	A Boolean expression which indicates whether the radial menu is expanded or collapsed.

By default, the Expanded property is False, which indicates that the radial menu is collapsed. The Expanded property indicates whether the radial menu is expanded or collapsed. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The State property specifies the state of the radial menu. The ParentCaption property specifies the caption to be displayed on the parent portion of the control, based on the radial menu's state. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state.

The following screen shot shows the control when it is collapsed:

The following screen shot shows the control when it is expanded:

property RadialMenu.ExtraCaption(Key as Variant, Property as PropertyLayerCaptionEnum) as Variant

Specifies any extra caption on the control.

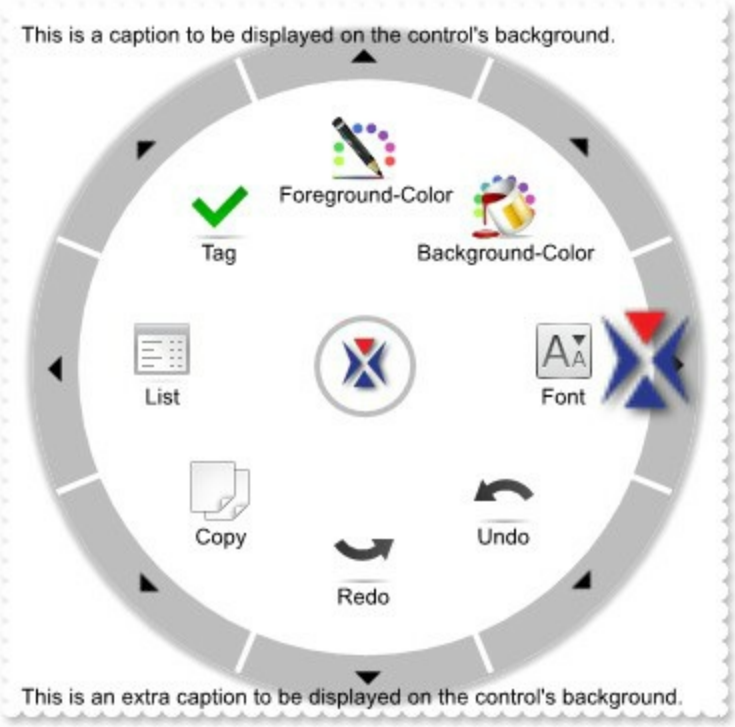
Type	Description
Key as Variant	A VARIANT expression that specifies the key of the extra caption. You can use any value to identify one extra caption.
Property as PropertyLayerCaptionEnum	A PropertyLayerCaptionEnum expression that specifies the extra caption's property to be changed.
Variant	A VARIANT expression that specifies the value of the extra caption's property.

The control support unlimited HTML captions to be place anywhere on the control or on any layer of the control. The Caption(exLayerCaption) specifies the HTML caption to be shown on the control/layer. The Images method specifies the list of icons the control can display. The HTMLPicture adds or replaces a picture in HTML captions. The Caption(exLayerCaptionBackgroundExt) property indicates unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the control / layer's background. The caption on the control stay on its position, no matter what layer is moved or rotated, while a caption on a layer gets moved or rotated together with the layer itself. The ForeColor property specifies the control's foreground color.

Any of the following properties can be used to display a HTML caption:

- Caption property specifies the caption to be shown on the control's foreground.
- ExtraCaption property specifies any extra caption to be shown on the control's foreground.

The following screen shot shows a few captions on the control's background:



The following samples show how you can display captions on the control's background:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   .MinVisibleCount = 6
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
   .Caption(0) = "This is a caption to be displayed on the control's background."
   .ExtraCaption("extra",0) = "This is an extra caption to be displayed on the control's background."
   .ExtraCaption("extra",3) = 2
   .ExtraCaption("extra",8) = True
   .ExtraCaption("extralogo",0) = "<img>logo:64</img>"
   .ExtraCaption("extralogo",3) = 4
   .ExtraCaption("extralogo",4) = "width-twidth"
   .EndUpdate
End With
```

VB6

```
With RadialMenu1
   .BeginUpdate
```

```
    .Expanded = True
    .MinVisibleCount = 6
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
    .Caption(exLayerCaption) = "This is a caption to be displayed on the control's
background."
    .ExtraCaption("extra",exLayerCaption) = "This is an extra caption to be displayed
on the control's background."
    .ExtraCaption("extra",exLayerCaptionAnchor) = 2
    .ExtraCaption("extra",exLayerCaptionWordWrap) = True
    .ExtraCaption("extralogo",exLayerCaption) = "<img>logo:64</img>"
    .ExtraCaption("extralogo",exLayerCaptionAnchor) = 4
    .ExtraCaption("extralogo",exLayerCaptionLeft) = "width-twidth"
    .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
    .BeginUpdate()
    .Expanded = True
    .MinVisibleCount = 6
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4"

.set_Caption(exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCaption,
 is a caption to be displayed on the control's background.")

.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exL
 is an extra caption to be displayed on the control's background.")

.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exL



.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.exL



.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnur
<img>logo:64</img>")
```

```
.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnur



.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLayerCaptionEnur
twidth")
   .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
   .BeginUpdate()
   .Expanded = True
   .MinVisibleCount = 6
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
   .set_Caption(EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCaption,"This
is a caption to be displayed on the control's background.")

.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCap
 is an extra caption to be displayed on the control's background.")

.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCap



.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCap



.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayer
<img>logo:64</img>")

.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayer



.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCaptionEnum.exLayer
twidth")
   .EndUpdate()
```

```
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->PutMinVisibleCount(6);
spRadialMenu1->GetItems()->PutToString(L"Item 1,Item 2,Item 3,Item 4");
spRadialMenu1->PutCaption(EXRADIALMENULib::exLayerCaption,"This is a caption
to be displayed on the control's background.");
spRadialMenu1->PutExtraCaption("extra",EXRADIALMENULib::exLayerCaption,"This
is an extra caption to be displayed on the control's background.");
spRadialMenu1-
>PutExtraCaption("extra",EXRADIALMENULib::exLayerCaptionAnchor,long(2));
spRadialMenu1-
>PutExtraCaption("extra",EXRADIALMENULib::exLayerCaptionWordWrap,VARIANT_TI

spRadialMenu1-
>PutExtraCaption("extralogo",EXRADIALMENULib::exLayerCaption,"
<img>logo:64</img>");
spRadialMenu1-
>PutExtraCaption("extralogo",EXRADIALMENULib::exLayerCaptionAnchor,long(4));
spRadialMenu1-
>PutExtraCaption("extralogo",EXRADIALMENULib::exLayerCaptionLeft,"width-
twidth");
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
RadialMenu1->MinVisibleCount = 6;
RadialMenu1->Items->ToString = L"Item 1,Item 2,Item 3,Item 4";
RadialMenu1-
>Caption[Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLayerCaption] =
TVariant("This is a caption to be displayed on the control's background.");
RadialMenu1-
>ExtraCaption[TVariant("extra"),Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLa
 = TVariant("This is an extra caption to be displayed on the control's background.");
RadialMenu1-
>ExtraCaption[TVariant("extra"),Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLa
 = TVariant(2);
RadialMenu1-
>ExtraCaption[TVariant("extra"),Exradialmenulib_tlb::PropertyLayerCaptionEnum::exLa
 = TVariant(true);
RadialMenu1-
>ExtraCaption[TVariant("extralogo"),Exradialmenulib_tlb::PropertyLayerCaptionEnum:
 = TVariant("<img>logo:64</img>");
RadialMenu1-
>ExtraCaption[TVariant("extralogo"),Exradialmenulib_tlb::PropertyLayerCaptionEnum:
 = TVariant(4);
RadialMenu1-
>ExtraCaption[TVariant("extralogo"),Exradialmenulib_tlb::PropertyLayerCaptionEnum:
 = TVariant("width-twidth");
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exradialmenu1.MinVisibleCount = 6;
exradialmenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4";
exradialmenu1.set_Caption(exontrol.EXRADIALMENULib.PropertyLayerCaptionEnum.e
 is a caption to be displayed on the control's background.");
```

```
exradialmenu1.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCa
 is an extra caption to be displayed on the control's background.");
exradialmenu1.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCa

exradialmenu1.set_ExtraCaption("extra",exontrol.EXRADIALMENULib.PropertyLayerCa

exradialmenu1.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLay
<img>logo:64</img>");
exradialmenu1.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLay

exradialmenu1.set_ExtraCaption("extralogo",exontrol.EXRADIALMENULib.PropertyLay
twidth");
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.Expanded = true;
   RadialMenu1.MinVisibleCount = 6;
   RadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4";
   RadialMenu1.Caption(0) = "This is a caption to be displayed on the control's
background.";
   RadialMenu1.ExtraCaption("extra",0) = "This is an extra caption to be displayed
on the control's background.";
   RadialMenu1.ExtraCaption("extra",3) = 2;
   RadialMenu1.ExtraCaption("extra",8) = true;
   RadialMenu1.ExtraCaption("extralogo",0) = "<img>logo:64</img>";
   RadialMenu1.ExtraCaption("extralogo",3) = 4;
   RadialMenu1.ExtraCaption("extralogo",4) = "width-twidth";
```

```
    RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With RadialMenu1
        .BeginUpdate
        .Expanded = True
        .MinVisibleCount = 6
        .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
        .Caption(0) = "This is a caption to be displayed on the control's background."
        .ExtraCaption("extra",0) = "This is an extra caption to be displayed on the
control's background."
        .ExtraCaption("extra",3) = 2
        .ExtraCaption("extra",8) = True
        .ExtraCaption("extralogo",0) = "<img>logo:64</img>"
        .ExtraCaption("extralogo",3) = 4
        .ExtraCaption("extralogo",4) = "width-twidth"
        .EndUpdate
    End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
```

```
axRadialMenu1.MinVisibleCount = 6;
axRadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4";
axRadialMenu1.set_Caption(EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC
 is a caption to be displayed on the control's background.");
axRadialMenu1.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEn
 is an extra caption to be displayed on the control's background.");
axRadialMenu1.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEn

axRadialMenu1.set_ExtraCaption("extra",EXRADIALMENULib.PropertyLayerCaptionEn

axRadialMenu1.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCapti
<img>logo:64</img>");
axRadialMenu1.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCapti

axRadialMenu1.set_ExtraCaption("extralogo",EXRADIALMENULib.PropertyLayerCapti
twidth");
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    ;

    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.Expanded(true);
    exradialmenu1.MinVisibleCount(6);
    exradialmenu1.Items().ToString("Item 1,Item 2,Item 3,Item 4");
    exradialmenu1.Caption(0/*exLayerCaption*/,"This is a caption to be displayed on
the control's background.");
    exradialmenu1.ExtraCaption("extra",0/*exLayerCaption*/,"This is an extra caption
to be displayed on the control's background.");

exradialmenu1.ExtraCaption("extra",3/*exLayerCaptionAnchor*/,COMVariant::createFr
```

```
exradialmenu1.ExtraCaption("extra",8/*exLayerCaptionWordWrap*/,COMVariant::creat

   exradialmenu1.ExtraCaption("extralogo",0/*exLayerCaption*/,"
<img>logo:64</img>");

exradialmenu1.ExtraCaption("extralogo",3/*exLayerCaptionAnchor*/,COMVariant::crea

   exradialmenu1.ExtraCaption("extralogo",4/*exLayerCaptionLeft*/,"width-twidth");
   exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   MinVisibleCount := 6;
   Items.ToString := 'Item 1,Item 2,Item 3,Item 4';
   set_Caption(EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCaption,'This
is a caption to be displayed on the control''s background.');

set_ExtraCaption('extra',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCapti
 is an extra caption to be displayed on the control''s background.');

set_ExtraCaption('extra',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCapti



set_ExtraCaption('extra',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerCapti



set_ExtraCaption('extralogo',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC



set_ExtraCaption('extralogo',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC
```

```
set_ExtraCaption('extralogo',EXRADIALMENULib.PropertyLayerCaptionEnum.exLayerC
twidth');
   EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   MinVisibleCount := 6;
   Items.ToString := 'Item 1,Item 2,Item 3,Item 4';
   Caption[EXRADIALMENULib_TLB.exLayerCaption] := 'This is a caption to be
displayed on the control''s background.';
   ExtraCaption['extra',EXRADIALMENULib_TLB.exLayerCaption] := 'This is an extra
caption to be displayed on the control''s background.';
   ExtraCaption['extra',EXRADIALMENULib_TLB.exLayerCaptionAnchor] :=
OleVariant(2);
   ExtraCaption['extra',EXRADIALMENULib_TLB.exLayerCaptionWordWrap] :=
OleVariant(True);
   ExtraCaption['extralogo',EXRADIALMENULib_TLB.exLayerCaption] :=
'<img>logo:64</img>';
   ExtraCaption['extralogo',EXRADIALMENULib_TLB.exLayerCaptionAnchor] :=
OleVariant(4);
   ExtraCaption['extralogo',EXRADIALMENULib_TLB.exLayerCaptionLeft] := 'width-
twidth';
   EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
   .BeginUpdate
   .Expanded = .T.
   .MinVisibleCount = 6
```

```
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4"
   .Object.Caption(0) = "This is a caption to be displayed on the control's
background."
   .Object.ExtraCaption("extra",0) = "This is an extra caption to be displayed on the
control's background."
   .Object.ExtraCaption("extra",3) = 2
   .Object.ExtraCaption("extra",8) = .T.
   .Object.ExtraCaption("extralogo",0) = "<img>logo:64</img>"
   .Object.ExtraCaption("extralogo",3) = 4
   .Object.ExtraCaption("extralogo",4) = "width-twidth"
   .EndUpdate
endwith
```

dBASE Plus

```
local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.MinVisibleCount = 6
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4"
oRadialMenu.Template = [Caption(0) = "This is a caption to be displayed on the
control's background."] // oRadialMenu.Caption(0) = "This is a caption to be
displayed on the control's background."
oRadialMenu.Template = [ExtraCaption("extra",0) = "This is an extra caption to be
displayed on the control's background."] // oRadialMenu.ExtraCaption("extra",0) =
"This is an extra caption to be displayed on the control's background."
oRadialMenu.Template = [ExtraCaption("extra",3) = 2] //
oRadialMenu.ExtraCaption("extra",3) = 2
oRadialMenu.Template = [ExtraCaption("extra",8) = True] //
oRadialMenu.ExtraCaption("extra",8) = true
oRadialMenu.Template = [ExtraCaption("extralogo",0) = "<img>logo:64</img>"] //
oRadialMenu.ExtraCaption("extralogo",0) = "<img>logo:64</img>"
oRadialMenu.Template = [ExtraCaption("extralogo",3) = 4] //
oRadialMenu.ExtraCaption("extralogo",3) = 4
oRadialMenu.Template = [ExtraCaption("extralogo",4) = "width-twidth"] //
```

```
oRadialMenu.ExtraCaption("extralogo",4) = "width-twidth"
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = .t.
oRadialMenu.MinVisibleCount = 6
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4"
oRadialMenu.Template = "Caption(0) = `This is a caption to be displayed on the control's background.`" // oRadialMenu.Caption(0) = "This is a caption to be displayed on the control's background."
oRadialMenu.Template = "ExtraCaption(`extra`,0) = `This is an extra caption to be displayed on the control's background.`" // oRadialMenu.ExtraCaption("extra",0) = "This is an extra caption to be displayed on the control's background."
oRadialMenu.Template = "ExtraCaption(`extra`,3) = 2" // oRadialMenu.ExtraCaption("extra",3) = 2
oRadialMenu.Template = "ExtraCaption(`extra`,8) = True" // oRadialMenu.ExtraCaption("extra",8) = .t.
oRadialMenu.Template = "ExtraCaption(`extralogo`,0) = `<img>logo:64</img>`" // oRadialMenu.ExtraCaption("extralogo",0) = "<img>logo:64</img>"
oRadialMenu.Template = "ExtraCaption(`extralogo`,3) = 4" // oRadialMenu.ExtraCaption("extralogo",3) = 4
oRadialMenu.Template = "ExtraCaption(`extralogo`,4) = `width-twidth`" // oRadialMenu.ExtraCaption("extralogo",4) = "width-twidth"
oRadialMenu.EndUpdate()
```

Visual Objects

```

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:MinVisibleCount := 6
```

```
oDCOCX_Exontrol1:Items:ToString := "Item 1,Item 2,Item 3,Item 4"
oDCOCX_Exontrol1:[Caption,exLayerCaption] := "This is a caption to be displayed on the control's background."
oDCOCX_Exontrol1:[ExtraCaption,"extra",exLayerCaption] := "This is an extra caption to be displayed on the control's background."
oDCOCX_Exontrol1:[ExtraCaption,"extra",exLayerCaptionAnchor] := 2
oDCOCX_Exontrol1:[ExtraCaption,"extra",exLayerCaptionWordWrap] := true
oDCOCX_Exontrol1:[ExtraCaption,"extralogo",exLayerCaption] := "<img>logo:64</img>"
oDCOCX_Exontrol1:[ExtraCaption,"extralogo",exLayerCaptionAnchor] := 4
oDCOCX_Exontrol1:[ExtraCaption,"extralogo",exLayerCaptionLeft] := "width-twidth"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.MinVisibleCount = 6
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4"
oRadialMenu.Caption(0,"This is a caption to be displayed on the control's background.")
oRadialMenu.ExtraCaption("extra",0,"This is an extra caption to be displayed on the control's background.")
oRadialMenu.ExtraCaption("extra",3,2)
oRadialMenu.ExtraCaption("extra",8,true)
oRadialMenu.ExtraCaption("extralogo",0,"<img>logo:64</img>")
oRadialMenu.ExtraCaption("extralogo",3,4)
oRadialMenu.ExtraCaption("extralogo",4,"width-twidth")
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
```

```
    Forward Send OnCreate
    Send ComBeginUpdate
    Set ComExpanded to True
    Set ComMinVisibleCount to 6
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Set ComToString of hoItems to "Item 1,Item 2,Item 3,Item 4"
    Send Destroy to hoItems
    Set ComCaption OLEexLayerCaption to "This is a caption to be displayed on the
control's background."
    Set ComExtraCaption "extra" OLEexLayerCaption to "This is an extra caption to be
displayed on the control's background."
    Set ComExtraCaption "extra" OLEexLayerCaptionAnchor to 2
    Set ComExtraCaption "extra" OLEexLayerCaptionWordWrap to True
    Set ComExtraCaption "extralogo" OLEexLayerCaption to "<img>logo:64</img>"
    Set ComExtraCaption "extralogo" OLEexLayerCaptionAnchor to 4
    Set ComExtraCaption "extralogo" OLEexLayerCaptionLeft to "width-twidth"
    Send ComEndUpdate
End_Procedure
```

XBase++

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

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

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

```
   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
      oRadialMenu:MinVisibleCount := 6
      oRadialMenu:Items():ToString := "Item 1,Item 2,Item 3,Item 4"
      oRadialMenu:SetProperty("Caption",0/*exLayerCaption*/,"This is a caption to be displayed on the control's background.")
      oRadialMenu:SetProperty("ExtraCaption","extra",0/*exLayerCaption*/,"This is an extra caption to be displayed on the control's background.")
      oRadialMenu:SetProperty("ExtraCaption","extra",3/*exLayerCaptionAnchor*/,2)

oRadialMenu:SetProperty("ExtraCaption","extra",8/*exLayerCaptionWordWrap*/,.T.)
      oRadialMenu:SetProperty("ExtraCaption","extralogo",0/*exLayerCaption*/,"<img>logo:64</img>")

oRadialMenu:SetProperty("ExtraCaption","extralogo",3/*exLayerCaptionAnchor*/,4)

oRadialMenu:SetProperty("ExtraCaption","extralogo",4/*exLayerCaptionLeft*/,"width-twidth")
      oRadialMenu:EndUpdate()

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

property RadialMenu.Float as RadialMenuFloatEnum

Specifies whether the control is shown as float.

Type	Description
RadialMenuFloatEnum	A RadialMenuFloatEnum expression that specifies whether the control is displayed on a form or on the screen.

By default, the Float property is exRadialMenuChild. The Float property specifies whether the control is shown as float. The AllowMoveOnFloat property has effect only if the control's Float property is exRadialMenuFloat or exRadialMenuFloatTopmost. The LayerUpdate property specifies where the control updates its content.

In order to make your eXRadialMenu control to display a popup/widget, (no form behind or form transparent), you need to use the following properties:

- **Float** property of the control, specifies whether the control is shown as float. You can use the Float property on exRadialMenuFloat or exRadialMenuFloatTopmost, to display the control as float (places the control above all non-topmost windows)

The setup installs the C:\Program Files\Exontrol\ExRadialMenu\Sample\VB\Float or C:\Program Files\Exontrol\ExRadialMenu\Sample\VC\Float that shows all these working.

In order to make your eXRadialMenu control to display a widget, (no form behind or form transparent), you need to use the following properties:

- Change the LayerUpdate property to **exLayerUpdateScreeen**, so the entire control is shown individually on the screen, with no form behind.

In order to make your eXRadialMenu library to display a transparent-control inside your form/dialog/window/child, you need to use the following properties:

- **LayerUpdate** property of the control, indicates where the control's content is updated. By default, the LayerUpdate property property is exLayerUpdateControl, which indicates that the control's content is shown on the control itself (no effect). If the LayerUpdate property is exLayerUpdateParent, the control does not show its background on the form that hosts it.

- You need to add **<supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>**, to your manifest file as follows. The transparent-eXRadialMenu as a child of your form, it is supported on Windows 8, and later.

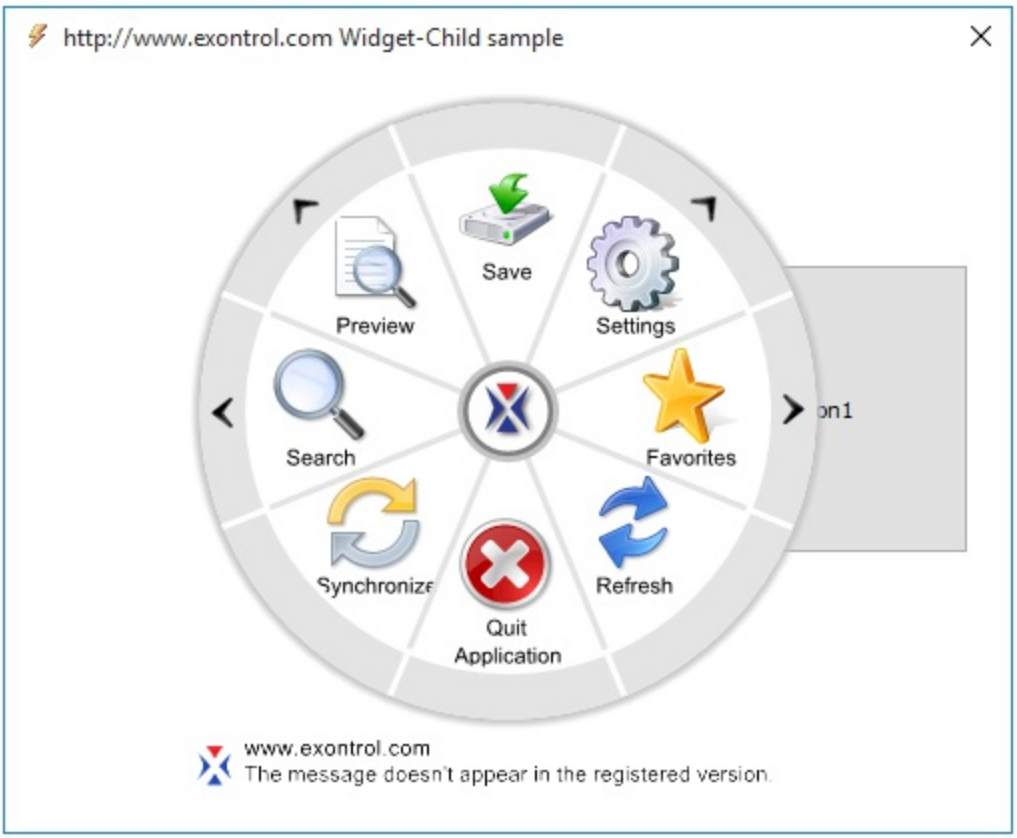
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1"
```

```
xmlns:asmv3="urn:schemas-microsoft-com:asm.v3">
   <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
      <application>
         <supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
      </application>
   </compatibility>
</assembly>
```

The setup installs the C:\Program Files\Exontrol\ExRadialMenu\Sample\VC\Widget-Child sample that shows all these working.

The following screen shot shows the control on a transparent form (**exLayerUpdateScreeen**):

The following screen shot shows the transparent-control on form (**exLayerUpdateParent**):


http://www.exontrol.com Widget-Child sample
Save
Preview
Settings
Search
Favorites
Synchronize
Refresh
Quit
Application
on1
www.exontrol.com
The message doesn't appear in the registered version.

property RadialMenu.Font as IFontDisp

Retrieves or sets the control's font.

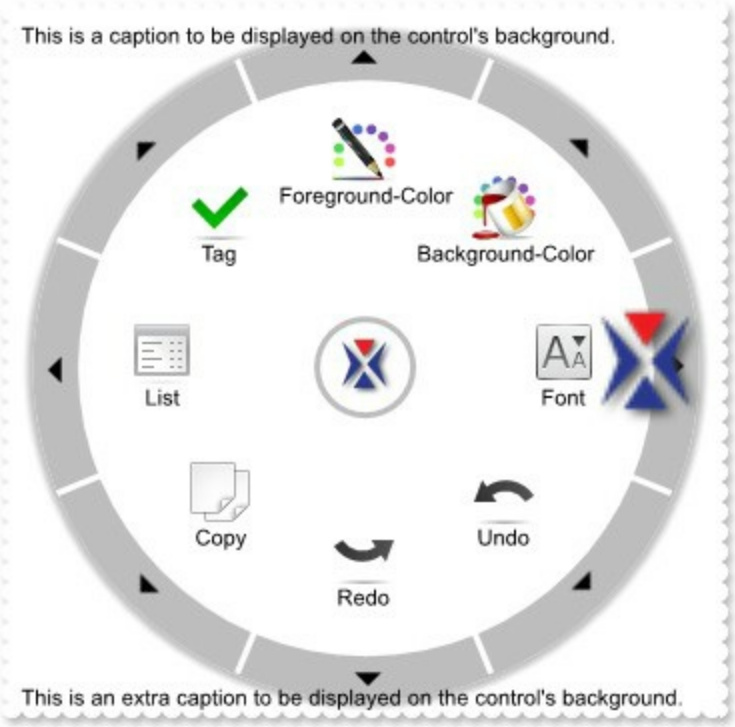
Type	Description
IFontDisp	A Font object used to paint the items.

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

Any of the following properties can be used to display a HTML caption:

- Caption property specifies the caption to be shown on the control's foreground.
- ExtraCaption property specifies any extra caption to be shown on the control's foreground.

The following screen shot shows a few captions on the control's background:



property RadialMenu.ForeColor as Color

Specifies the control's foreground color.

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

By default, the ForeColor property is system window text color. The ForeColor property specifies the foreground color for all items with ForeColor property on -1 (by default). The ForeColor property specifies the item's foreground color. The Caption property retrieves or sets a value that indicates the item's caption. You can specify the caption of the item using the Caption parameter of the Add method. The Image property assigns an icon/picture to the item.

method RadialMenu.FormatABC (Expression as String, [A as Variant], [B as Variant], [C as Variant])

Formats the A,B,C values based on the giving expression and returns the result.

Type	Description
Expression as String	A String that defines the expression to be evaluated.
A as Variant	A VARIANT expression that indicates the value of the A keyword.
B as Variant	A VARIANT expression that indicates the value of the B keyword.
C as Variant	A VARIANT expression that indicates the value of the C keyword.

Return	Description
Variant	A VARIANT expression that indicates the result of the evaluation the Expression.

The FormatABC method formats the A,B,C values based on the giving expression and returns the result. The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

For instance:

- "A + B + C", adds / concatenates the values of the A, B and C
- "value MIN 0 MAX 99", limits the value between 0 and 99
- "value format ``", formats the value with two decimals, according to the control's panel setting
- "date(`now`)" returns the current time as double

The Expression of the FormatABC method supports the following keywords, constants, operators and functions:

- **A** or **value** keyword, indicates a variable A whose value is giving by the A parameter
- **B** keyword, indicates a variable B whose value is giving by the B parameter
- **C** keyword, indicates a variable C whose value is giving by the C parameter

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum

value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns

always a value greater than 10.

- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.

- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the

result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

property RadialMenu.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

Type	Description
New as Boolean	A Boolean expression that indicates whether to specify the anchors never clicked or anchors being clicked.
String	A String expression that indicates the HTMLformat to apply to anchor elements.

By default, the FormatAnchor(**True**) property is "<u><fgcolor=0000FF>#" that indicates that the anchor elements (that were never clicked) are underlined and shown in light blue. Also, the FormatAnchor(**False**) property is "<u><fgcolor=000080>#" that indicates that the anchor elements are underlined and shown in dark blue. The visual effect is applied to the anchor elements, if the FormatAnchor property is not empty. For instance, if you want to do not show with a new effect the clicked anchor elements, you can use the FormatAnchor(**False**) = "", that means that the clicked or not-clicked anchors are shown with the same effect that's specified by FormatAnchor(**True**). An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the AnchorClick event to notify that the user clicks an anchor element. This event is fired only if prior clicking the control it shows the hand cursor. The AnchorClick event carries the identifier of the anchor, as well as application options that you can specify in the anchor element. The hand cursor is shown when the user hovers the mouse on the anchor elements.

method RadialMenu.GoBack ()

Advances to the parent item.

Type	Description

property RadialMenu.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 relative path to the picture file, being loaded. The PicturesPath specifies the path to load the pictures from. • 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 Exontrol's 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).

By default, the control loads the following predefined pictures:

- "arrow"  , which is usually displayed on the sub-items portion of the control, while there is any child-items or a custom control attached.
- "logo",  , which is usually displayed on the parent portion of the control, when the

control is collapsed .

- "goback",  , which is usually displayed on the parent portion of the control, when the control is expanded.

The following sample shows how to put a custom size picture in the column's header:

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

<COLUMN1>.HTMLCaption = "A <img>pic1</img>"
<COLUMN2>.HTMLCaption = "B <img>pic2</img>"
<COLUMN3>.HTMLCaption = "A <img>pic1</img> + B <img>pic2</img>"
```

property RadialMenu.hWnd as Long

Retrieves the control's window handle.

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

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

method RadialMenu.Images (Handle as Variant)

Sets a runtime the control's image tree.

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.

property RadialMenu.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.

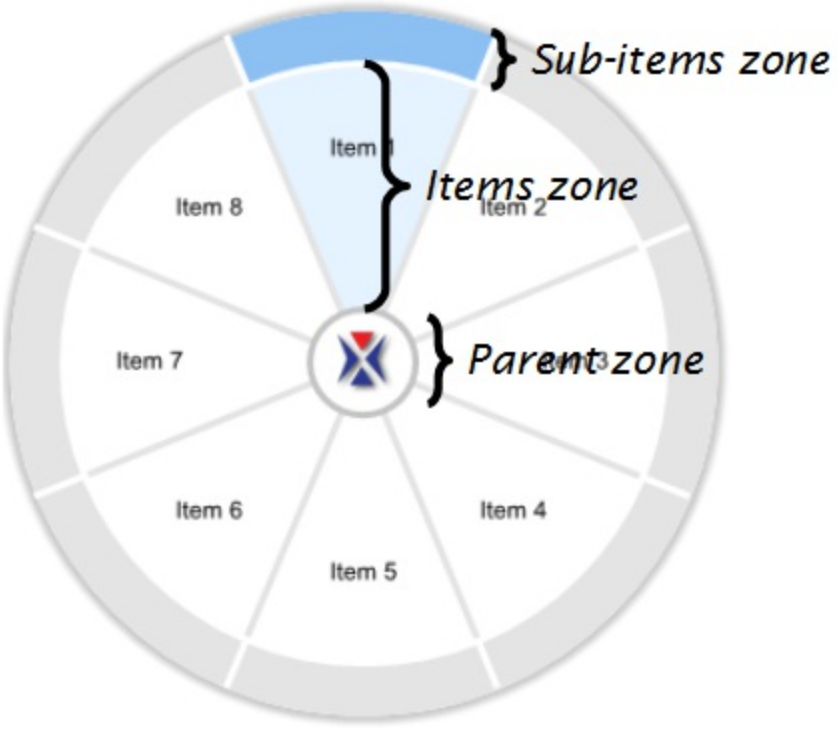
property RadialMenu.IndexFromPoint (Type as RadialItemsEnum, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

Retrieves the index of the radial pie, from the point.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies whether the items/sub-items/any zone is queried for an item
X as OLE_XPOS_PIXELS	A long expression that specifies the x-position in client coordinate to get the index of the radial pie / slice from
Y as OLE_YPOS_PIXELS	A long expression that specifies the y-position in client coordinate to get the index of the radial pie / slice from
Long	A Long expression that specifies the zero-based index of the radial pie, from the point. If -1, no index is found.

The IndexFromPoint property gets the index of the radial pie / slice from the cursor. The ItemFromPoint property gets the item from the cursor. The ParentOnPoint property indicates if the point hits the parent zone of the radial menu. The AnchorFromPoint property retrieves the identifier of the anchor from point. The MouseMove event is generated continually as the mouse pointer moves across the control.

The following screen shot, shows the portions/parts/zones of the radial menu:



property RadialMenu.InflateCustom as String

Inflates or deflates the client area of the custom portion of the control.

Type	Description
String	A String expression that inflates or deflates the client area of the custom portion of the control.

The InflateCustom property inflates or deflates the client area of the custom portion of the control. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control. The InflateItems property Inflates or deflates the client area of the items portion of the control. The InflateParentPicture property inflates or deflates the client area to display the picture on the background of the parent's zone of the control. The InflateRadialMenu property inflates or deflates the client area of the radial menu control. The ParentSize property specifies the size to display the parent zone. The ParentSize property specifies the size to display the parent zone. The SubItemsSize property specifies the size to display the sub-items zone.

The InflateCustom property supports the following keywords:

- **value**, indicates the radius in pixels, of the items/custom section of the control.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero

if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the

ltrim(" mihai") returns "mihai"

- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001*

13:00#) returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.InflateItems as String

Inflates or deflates the client area of the items portion of the control.

Type	Description
String	A String expression that inflates or deflates the client area of the items portion of the control.

By default, The InflateItems property is "-0*dpi". The InflateItems property Inflates or deflates the client area of the items portion of the control. The InflateRadialMenu property inflates or deflates the client area of the radial menu control. The InflateCustom property inflates or deflates the client area of the custom portion of the control. The InflateParentPicture property inflates or deflates the client area to display the picture on the background of the parent's zone of the control. The ParentSize property specifies the size to display the parent zone. The ParentSize property specifies the size to display the parent zone. The SubItemsSize property specifies the size to display the sub-items zone.

The InflateItems property supports the following keywords:

- **value**, indicates the radius in pixels, of the items section of the control.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54

- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the

trim(" mihai ") returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"

- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.InflateParentPicture as String

Inflates or deflates the client area to display the picture on the background of the parent's zone of the control.

Type	Description
String	A String expression that inflates or deflates the client area to display the picture on the background of the parent's zone of the control.

By default, The InflateParentPicture property is "4*dpi", which indicates 4 pixels for a DPI 100%. By default, the parent picture is stretch on the parent zone of the control. The InflateParentPicture property inflates or deflates the client area to display the picture on the background of the parent's zone of the control. The ParentPicture property indicates the picture to be shown on the parent zone's background. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The ParentSize property specifies the size to display the parent zone.

The InflateParentPicture property supports the following keywords:

- **value**, indicates the radius in pixels, of the parent section of the control.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero

if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the

ltrim(" mihai") returns "mihai"

- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001*

13:00#) returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.InflateRadialMenu as String

Inflates or deflates the client area of the radial menu control.

Type	Description
String	A String expression that inflates or deflates the client area of the radial menu control.

By default, The InflateRadialMenu property is "0", which indicates no effect. The InflateRadialMenu property inflates or deflates the client area of the radial menu control. The InflateItems property Inflates or deflates the client area of the items portion of the control. The InflateCustom property inflates or deflates the client area of the custom portion of the control. The InflateParentPicture property inflates or deflates the client area to display the picture on the background of the parent's zone of the control. The ParentSize property specifies the size to display the parent zone. The ParentSize property specifies the size to display the parent zone. The SubItemsSize property specifies the size to display the sub-items zone.

The InflateRadialMenu property supports the following keywords:

- **value**, indicates the radius in pixels, of the control.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for

restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".

- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the

rtrim("mihai ") returns "mihai"

- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the

"MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

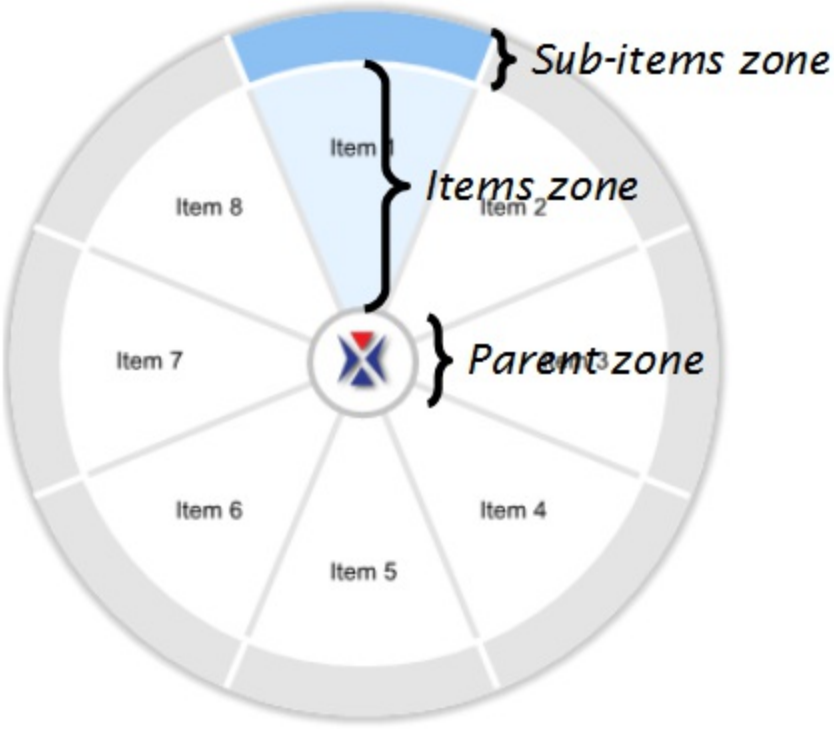
property RadialMenu.ItemFromPoint (Type as RadialItemsEnum, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Item

Retrieves the item, from the point.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies whether the items/sub-items/any zone is queried for an item
X as OLE_XPOS_PIXELS	A long expression that specifies the x-position in client coordinate to get the item from.
Y as OLE_YPOS_PIXELS	A long expression that specifies the y-position in client coordinate to get the item from.
Item	An Item object from the cursor, or nothing, if no item at the cursor position.

The ItemFromPoint property gets the item from the cursor. The ParentOnPoint property indicates if the point hits the parent zone of the radial menu. The IndexFromPoint property gets the item from the cursor. The AnchorFromPoint property retrieves the identifier of the anchor from point. The MouseMove event is generated continually as the mouse pointer moves across the control.

The following screen shot, shows the portions/parts/zones of the radial menu:



property RadialMenu.Items as Items

Retrieves the control's Items collection.

Type	Description
Items	A Items object that contains items of the control.

The Items property gives access to the control's Items collection.

The user can add new items to the control using any of the following:

- Add method, adds a new item to the control. The Add method can be used to add child-items as well.
- ToString property of the Items collection, loads or saves the Items collection using string representation.
- ToString property of the control, loads or saves the Items collection using string representation.

The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

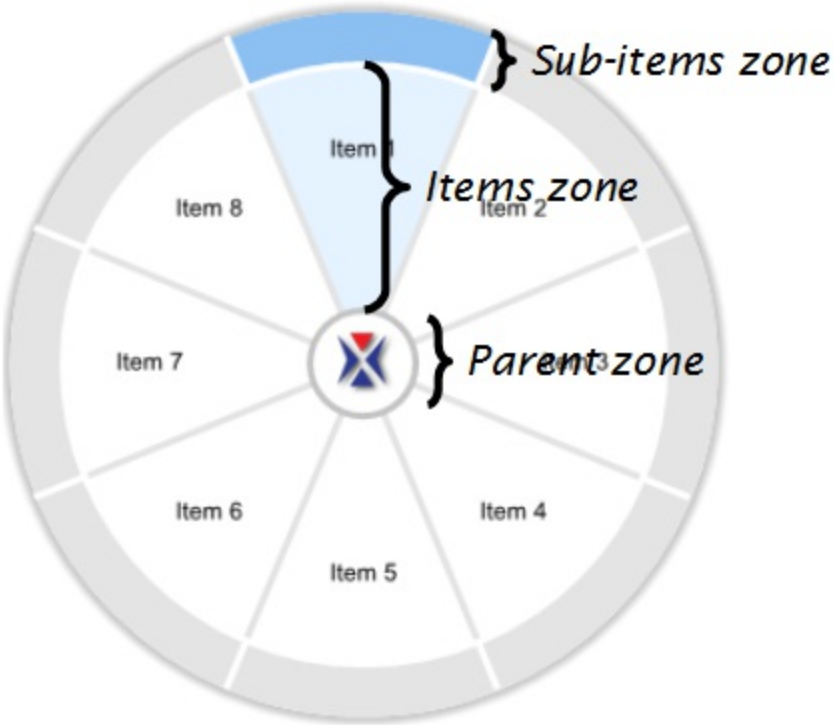
property RadialMenu.ItemsBackAlpha as Byte

Specifies the value of alpha / opacity channel to show the items portion of the radial menu.

Type	Description
Byte	A BYTE expression that specifies the value of alpha / opacity channel to show the items portion of the radial menu. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

By default, the The ItemsBackAlpha property is 255. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub- items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone.

The following screen shot, shows the portions/parts/zones of the radial menu:



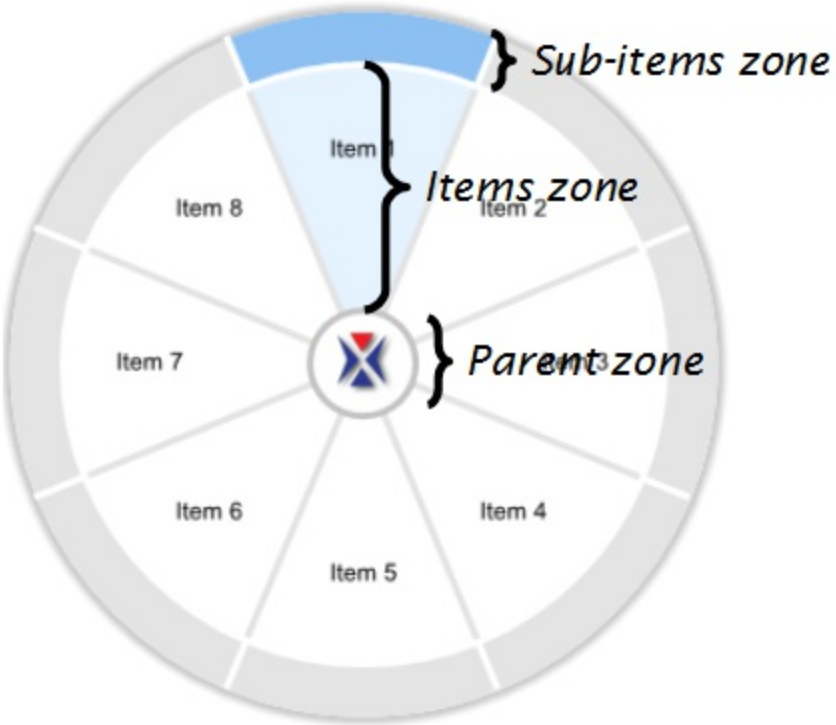
property RadialMenu.ItemsBackColor as Color

Specifies the color to show the items portion of the radial menu.

Type	Description
Color	A BYTE expression that specifies the value of alpha / opacity channel to show the items portion of the radial menu. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

By default, the The ItemsBackColor property is -1, so no background color is applied to the items section of the control. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub- items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone.

The following screen shot, shows the portions/parts/zones of the radial menu:



property RadialMenu.ItemsImageHeight(Type as RadialItemsEnum) as String

Specifies the height to display the item's image.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies the part of the item, to change the size of the image.
String	A String expression that defines the height to display the item's image.

By default, the ItemsImageHeight property is "pheight", which indicates that the item's image is displayed using its default size. The ItemsImageWidth / ItemsImageHeight property specifies the size to display the item's image. The Image property retrieves or sets a value that indicates the item's image.

The ItemsImageHeight property supports the following keywords:

- **pwidth**, indicates the width in pixels of the item's image
- **pheight**, indicates the height in pixels of the item's image
- **width**, indicates the width in pixels of the control.
- **height**, indicates the height in pixels of the control.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero

if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the

ltrim(" mihai") returns "mihai"

- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001*

13:00#) returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.ItemsImageWidth(Type as RadialItemsEnum) as String

Specifies the width to display the item's image.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that specifies the part of the item, to change the size of the image.
String	A String expression that defines the width to display the item's image.

By default, the ItemsImageWidth property is "pwidth", which indicates that the item's image is displayed using its default size. The ItemsImageWidth / ItemsImageHeight property specifies the size to display the item's image. The Image property retrieves or sets a value that indicates the item's image.

The ItemsImageWidth property supports the following keywords:

- **pwidth**, indicates the width in pixels of the item's image
- **pheight**, indicates the height in pixels of the item's image
- **width**, indicates the width in pixels of the control.
- **height**, indicates the height in pixels of the control.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero

if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the

ltrim(" mihai") returns "mihai"

- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001*

13:00#) returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.ItemsPicture as Variant

Indicates the picture to be shown on the items's background.

Type	Description

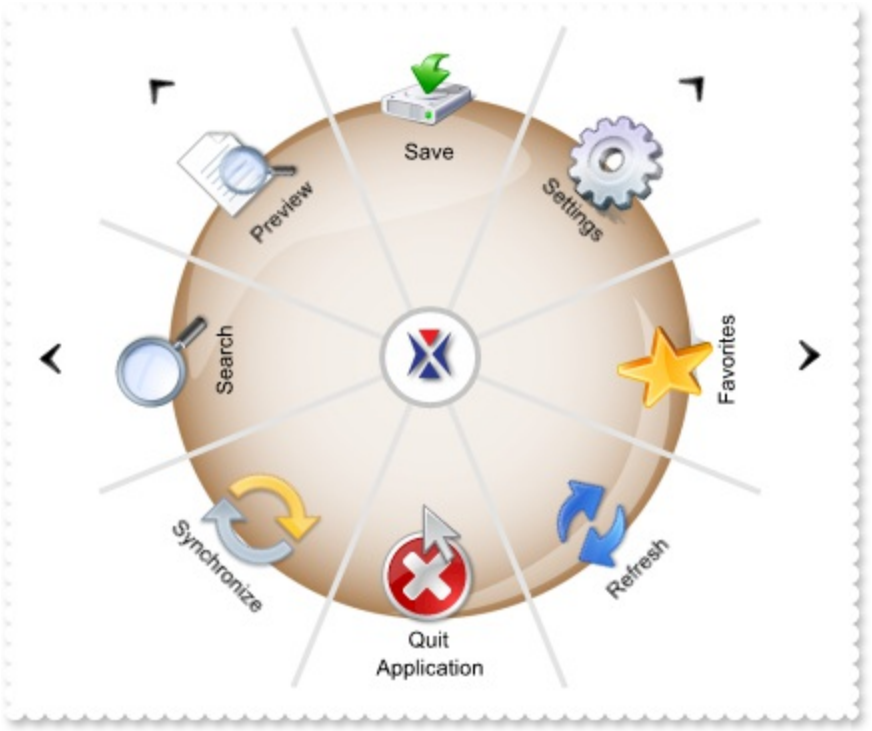
Variant	A VARIANT expression that specifies the icon/picture/image to be displayed as described:
	
	• A String expression indicates:
	  ◦ a name of a picture file in the PicturePath folder. For instance, ItemsPicture = "favorites.png", loads the favorites.png file if found in the PicturePath folder.
	  ◦ a picture file including its absolute path. For instance, ItemsPicture = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor loads the favorites.png file from absolute path
	  ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, ItemsPicture = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor
	  ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on.
	• A Picture object that indicates the picture to be displayed. For instance, ItemsPicture = LoadPicture("picture.jpg")
	
	If no picture/image is found, the items section displays no picture/image.

By default, The ItemsPicture property is empty. The ItemsPicture property indicates the picture to be shown on the items's background. The BackgroundPicture property indicates the picture to be shown on the radial menu's background. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone. The

ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu.

The following screen shot shows control with no background picture (default):

The following screen shot shows control with a background picture, for the items section:



property RadialMenu.LayerUpdate as LayerUpdateEnum

Specifies where the control updates its content.

Type	Description
LayerUpdateEnum	A LayerUpdateEnum expression that specifies where the control updates its content.

By default, the LayerUpdate property property is exLayerUpdateControl, which indicates that the control's content is shown on the control itself (no effect). The LayerUpdate property indicates where the control's content is updated. The control support transparent form, or in other words, displaying the control's itself without its form behind.

In order to make your eXRadialMenu control to display a popup/widget, (no form behind or form transparent), you need to use the following properties:

- **Float** property of the control, specifies whether the control is shown as float. You can use the Float property on exRadialMenuFloat or exRadialMenuFloatTopmost, to display the control as float (places the control above all non-topmost windows)

The setup installs the C:\Program Files\Exontrol\ExRadialMenu\Sample\VB\Float or C:\Program Files\Exontrol\ExRadialMenu\Sample\VC\Float that shows all these working.

In order to make your eXRadialMenu control to display a widget, (no form behind or form transparent), you need to use the following properties:

- Change the LayerUpdate property to **exLayerUpdateScreeen**, so the entire control is shown individually on the screen, with no form behind.

In order to make your eXRadialMenu library to display a transparent-control inside your form/dialog/window/child, you need to use the following properties:

- **LayerUpdate** property of the control, indicates where the control's content is updated. By default, the LayerUpdate property property is exLayerUpdateControl, which indicates that the control's content is shown on the control itself (no effect). If the LayerUpdate property is exLayerUpdateParent, the control does not show its background on the form that hosts it.

- You need to add **<supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>**, to your manifest file as follows. The transparent-eXRadialMenu as a child of your form, it is supported on Windows 8, and later.

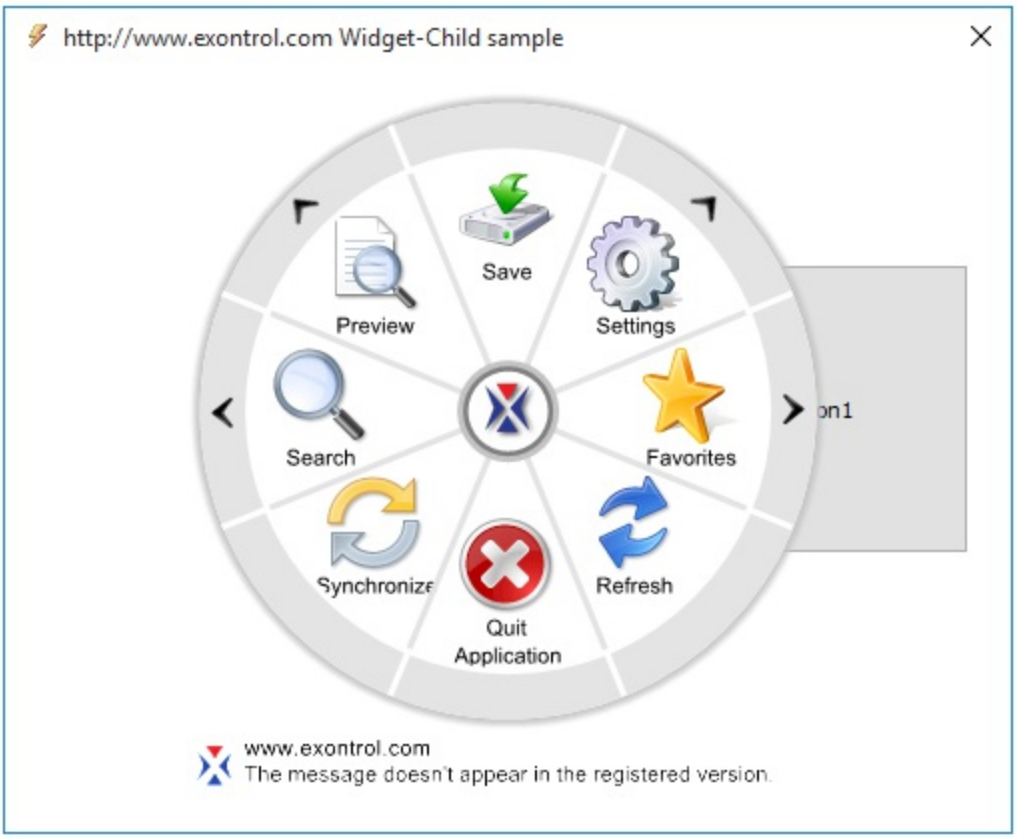
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1"
```

```
xmlns:asmv3="urn:schemas-microsoft-com:asm.v3">
   <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
      <application>
         <supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
      </application>
   </compatibility>
</assembly>
```

The setup installs the C:\Program Files\Exontrol\ExRadialMenu\Sample\VC\Widget-Child sample that shows all these working.

The following screen shot shows the control on a transparent form (**exLayerUpdateScreeen**):

The following screen shot shows the transparent-control on form (**exLayerUpdateParent**):


http://www.exontrol.com Widget-Child sample
Save
Preview
Settings
Search
Favorites
Synchronize
Refresh
Quit
Application
on1
www.exontrol.com
The message doesn't appear in the registered version.

property RadialMenu.MinVisibleCount as Long

Specifies the minimum number of items being visible on the radial menu.

Type	Description
Long	A Long expression that specifies minimum number of items being visible on the radial menu. The MinVisibleCount property can not be less than 3.

By default, the MinVisibleCount property is 8, which indicates that the control displays at least 8 slices, when it gets expanded. The Root.Items.Count property specifies the number of items that the control initially display when it gets expanded. The Add method, adds a new item to the control. The Clear method clears all items from the collection. The Item property accesses an item based on its index or name. The Remove method removes an item from the collection.

property RadialMenu.ParentBackAlpha as Byte

Specifies the value of alpha / opacity channel to show the parent portion of the radial menu.

Type	Description
Byte	A BYTE expression that specifies the value of alpha / opacity channel to show the parent portion of the radial menu. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

By default, The ParentBackAlpha property is 255. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The RadialLineColor(exRadialParentBorder) property specifies the color to show the border around the parent section of the control. The RadialLineColor(exRadialHotParent) property specifies the color to show the border around the parent section of the control, while the cursor hovers the parent section.

The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The ParentPicture property indicates the picture to be shown on the parent zone's background. The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions. The ParentPicture property indicates the picture to be shown on the parent zone's background. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu.

property RadialMenu.ParentBackColor as Color

Specifies the color to show the parent portion of the radial menu.

Type	Description
Color	A Color expression that specifies the color to show the parent portion of the radial menu. If -1, no solid color is applied on the parent portion of the control.

By default, The ParentBackColor property is system window's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The RadialLineColor(exRadialParentBorder) property specifies the color to show the border around the parent section of the control. The RadialLineColor(exRadialHotParent) property specifies the color to show the border around the parent section of the control, while the cursor hovers the parent section.

The ParentSize property specifies the size to display the parent zone. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The ParentPicture property indicates the picture to be shown on the parent zone's background. The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions. The ParentPicture property indicates the picture to be shown on the parent zone's background. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu.

property RadialMenu.ParentCaption(State as RadialMenuStateEnum) as String

Specifies the caption to be shown on the parent zone, based on the state of the radial menu.

Type	Description
State as RadialMenuStateEnum	A RadialMenuStateEnum expression that determines the state whose caption to be changed.
String	A String expression that specifies the HTML caption to be displayed on the parent portion of the control.

By default, the ParentCaption property is "". The ParentCaption property specifies the caption to be displayed on the parent portion of the control, based on the radial menu's state. The ParentSize property specifies the size to display the parent zone. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu. The ParentPicture property indicates the picture to be shown on the parent zone's background.

The following screen shot shows the parent with a different caption:

The ParentCaption property supports the following built-in HTML format:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text

- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to

stretch the picture, else the original size of the picture is used.

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

gradient-center

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

outlined

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

shadow

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

*</**sha**>*" gets:

property RadialMenu.ParentImage(State as RadialMenuStateEnum) as Variant

Specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu.

Type	Description
State as RadialMenuStateEnum	A RadialMenuStateEnum expression that determines the state whose image to be changed.
Variant	A VARIANT expression that specifies the icon/picture/image to be displayed as described: • A String expression indicates: ◦ a name of a picture file in the PicturePath folder. For instance, ParentImage = "favorites.png", loads the favorites.png file if found in the PicturePath folder. ◦ a picture file including its absolute path. For instance, ParentImage = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favo loads the favorites.png file from absolute path ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, ParentImage = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favo ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on. • A Picture object that indicates the picture to be displayed. For instance, ParentImage = LoadPicture("picture.jpg") • a long/string expression that specifies the index of the icon to be displayed (0-based). The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. If no icon/picture/image is found, the item displays no icon/picture/image.

The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The ParentSize property specifies the size to display the parent zone. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu. The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions. The ParentPicture property indicates the picture to be shown on the parent zone's background.

The ParentImage(exRadialMenuStateAll) changes the image for the parent portion of the control for all states of the radial menu control.

By default, the ParentImage property based on the state of the radial menu is:

- **exRadialMenuCollapsed**: "logo" indicates that the HTMLPicture("logo") is displayed when the control is collapsed.
- **exRadialMenuExpandedNoItems**: "logo" indicates that the HTMLPicture("logo") is displayed when the control is expanded, but it contains no child items.
- **exRadialMenuExpandedRootItem**: "logo" indicates that the HTMLPicture("logo") is displayed when the control is expanded, but it contains child items.
- **exRadialMenuExpandedChildItem**: "inherit", indicates that the item's Image property is displayed when a child-item is browsed, else if it is not found, it show the picture of exRadialMenuMissingInheritItemImage state.
- **exRadialMenuMissingInheritItemImage**: "goback" indicates that the HTMLPicture("goback") is displayed when the item's image is missing or not found.

property RadialMenu.ParentImageHeight(State as RadialMenuStateEnum) as String

Specifies the height to display the parent image in specified state.

Type	Description
State as RadialMenuStateEnum	A RadialMenuStateEnum expression that determines the state whose size to be changed.
String	A String expression that indicates the height of the image to be shown on the parent portion of the control.

By default, the ParentImageHeight property is "32*dpi", which indicates 32 pixels if DPI setting is 100%, or 48 pixels (32 * 1.5) if DPI setting is 150% . The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentSize property specifies the size to display the parent zone. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu. The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions. The ParentPicture property indicates the picture to be shown on the parent zone's background.

The ParentImageHeight(exRadialMenuStateAll) changes the width for the parent portion of the control for all states of the radial menu control.

The ParentImageHeight property supports the following keywords:

- **pwidth**, indicates the width in pixels of the picture
- **pheight**, indicates the height in pixels of the picture
- **width**, indicates the width in pixels of the parent portion of the control (ParentSize property).
- **height**, indicates the height in pixels of the parent portion of the control (ParentSize property).

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%,

the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.

- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI

- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if

the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.ParentImageWidth(State as RadialMenuStateEnum) as String

Specifies the width to display the parent image in specified state.

Type	Description
State as RadialMenuStateEnum	A RadialMenuStateEnum expression that determines the state whose size to be changed.
String	A String expression that indicates the width of the image to be shown on the parent portion of the control.

By default, the ParentImageWidth property is "32*dpi", which indicates 32 pixels if DPI setting is 100%, or 48 pixels (32 * 1.5) if DPI setting is 150% . The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentSize property specifies the size to display the parent zone. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu. The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions. The ParentPicture property indicates the picture to be shown on the parent zone's background.

The ParentImageWidth(exRadialMenuStateAll) changes the width for the parent portion of the control for all states of the radial menu control.

The ParentImageWidth property supports the following keywords:

- **pwidth**, indicates the width in pixels of the picture
- **pheight**, indicates the height in pixels of the picture
- **width**, indicates the width in pixels of the parent portion of the control (ParentSize property).
- **height**, indicates the height in pixels of the parent portion of the control (ParentSize property).

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%,

the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.

- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI

- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if

the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

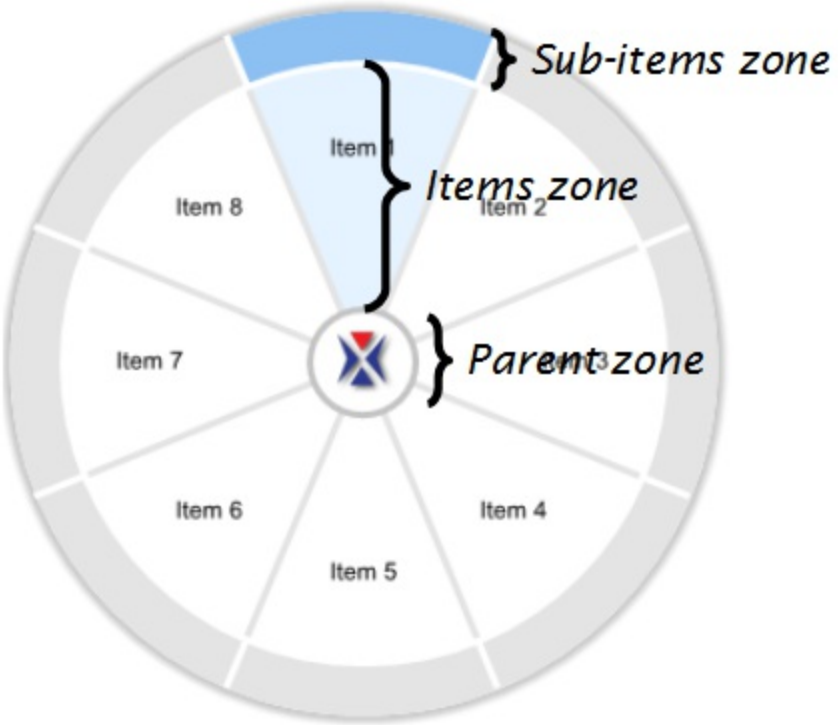
property RadialMenu.ParentOnPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Boolean

Indicates if the point hits the parent zone of the radial menu.

Type	Description
X as OLE_XPOS_PIXELS	A long expression that specifies the x-position in client coordinate to get the parent section from.
Y as OLE_YPOS_PIXELS	A long expression that specifies the y-position in client coordinate to get the parent section from.
Boolean	A Boolean expression that specifies whether the cursor hovers the parent section of the control.

The ParentOnPoint property indicates if the point hits the parent zone of the radial menu. The ItemFromPoint property gets the item from the cursor. The IndexFromPoint property gets the item from the cursor. The AnchorFromPoint property retrieves the identifier of the anchor from point. The MouseMove event is generated continually as the mouse pointer moves across the control.

The following screen shot, shows the portions/parts/zones of the radial menu:



property RadialMenu.ParentPicture as Variant

Indicates the picture to be shown on the parent zone's background.

Type	Description

Variant	A VARIANT expression that specifies the icon/picture/image to be displayed as described:
	
	• A String expression indicates:
	  ◦ a name of a picture file in the PicturePath folder. For instance, ParentPicture = "favorites.png", loads the favorites.png file if found in the PicturePath folder.
	  ◦ a picture file including its absolute path. For instance, ParentPicture = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor loads the favorites.png file from absolute path
	  ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, ParentPicture = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor
	  ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on.
	• A Picture object that indicates the picture to be displayed. For instance, ParentPicture = LoadPicture("picture.jpg")
	
	If no picture/image is found, the item displays no picture/image.

By default, The ParentPicture property is empty, which indicates that no picture is shown on the parent portion's background. By default, the parent picture is stretch on the parent zone of the control. The InflateParentPicture property inflates or deflates the client area to display the picture on the background of the parent's zone of the control. The ParentPicture property indicates the picture to be shown on the parent zone's background. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's

state. The ParentSize property specifies the size to display the parent zone. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu. The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions.

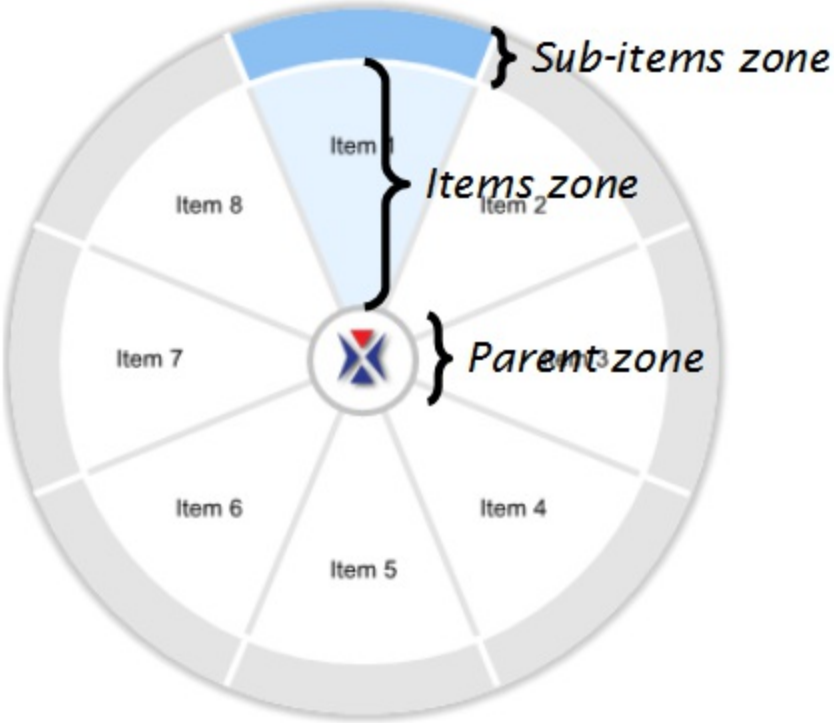
property RadialMenu.ParentSize as String

Specifies the size to display the parent zone.

Type	Description
String	A String expression that defines the size of the parent portion of the control.

By default, The ParentSize property is "24*dpi", which indicates 24 pixels if DPI setting is 100%, or 36 pixels (24 * 1.5) if DPI setting is 150% . The ParentSize property specifies the size to display the parent zone. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state. The ParentPicture property indicates the picture to be shown on the parent zone's background. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The RadialLineColor(exRadialParentBorder) property specifies the color to show the border around the parent section of the control. The RadialLineColor(exRadialHotParent) property specifies the color to show the border around the parent section of the control, while the cursor hovers the parent section. The ParentPicture property indicates the picture to be shown on the parent zone's background. The SubItemsSize property specifies the size to display the sub-items zone. The InflateRadialMenu property inflates or deflates the client area of the radial menu control. The InflateItems property Inflates or deflates the client area of the items portion of the control. The InflateCustom property inflates or deflates the client area of the custom portion of the control. The InflateParentPicture property inflates or deflates the client area to display the picture on the background of the parent's zone of the control. The ParentSize property specifies the size to display the parent zone.

The following screen shot, shows the portions/parts/zones of the radial menu:



The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions. The ParentPicture property indicates the picture to be shown on the parent zone's background. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The State property specifies the state of the radial menu.

The ParentSize property supports the following keywords:

- **value**, indicates the radius in pixels, of the radial menu.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if

current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

variable := expression

where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance,

the *upper("mihai")* returns "MIHAI"

- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"

- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.PicturesPath as String

Specifies the path to load the pictures from.

Type	Description
String	A String expression that defines the folder to load pictures from.

By default, the PicturesPath property is empty. The PicturesPath property specifies the path to load the pictures from. The Image property specifies the item's image. The Images method loads icons into the control. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture adds or replaces a picture in HTML captions.

The following sample show how you can load items /images to the control:

VBA (MS Access, Excell...)

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .HTMLPicture("arrow") = "arrow.png"
   .SubItemsBackColor = RGB(190,190,190)
   .ShadowColor = .SubItemsBackColor
   .RadialLineColor(6) = .SubItemsBackColor
   .RadialLineColor(5) = -1
   With .Items
      .Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
      .Add("Background-Color","color_fill.png").Items.ToString = "Background"
      .Add("Font","format_font_size_less.png").Items.ToString = "Font"
      .Add("Undo","edit_undo.png").Items.ToString = "Undo"
      .Add("Redo","edit_redo.png").Items.ToString = "Redo"
      .Add("Copy","edit_copy.png").Items.ToString = "Copy"
      .Add("List","fileview_text.png").Items.ToString = "List"
      .Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
   End With
   .EndUpdate
End With
```

VB6

```
With RadialMenu1
   .BeginUpdate
   .Expanded = True
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .HTMLPicture("arrow") = "arrow.png"
   .SubItemsBackColor = RGB(190,190,190)
   .ShadowColor = .SubItemsBackColor
   .RadialLineColor(exRadialParentBorder) = .SubItemsBackColor
   .RadialLineColor(exRadialItemsGridLines) = -1
   With .Items
      .Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
      .Add("Background-Color","color_fill.png").Items.ToString = "Background"
      .Add("Font","format_font_size_less.png").Items.ToString = "Font"
      .Add("Undo","edit_undo.png").Items.ToString = "Undo"
      .Add("Redo","edit_redo.png").Items.ToString = "Redo"
      .Add("Copy","edit_copy.png").Items.ToString = "Copy"
      .Add("List","fileview_text.png").Items.ToString = "List"
      .Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
   End With
   .EndUpdate
End With
```

VB.NET

```
With Exradialmenu1
   .BeginUpdate()
   .Expanded = True
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .set_HTMLPicture("arrow","arrow.png")
   .SubItemsBackColor = Color.FromArgb(190,190,190)
   .ShadowColor = .SubItemsBackColor

.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialParentBorde



.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialItemsGrid
```

```
    With .Items
        .Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
        .Add("Background-Color","color_fill.png").Items.ToString = "Background"
        .Add("Font","format_font_size_less.png").Items.ToString = "Font"
        .Add("Undo","edit_undo.png").Items.ToString = "Undo"
        .Add("Redo","edit_redo.png").Items.ToString = "Redo"
        .Add("Copy","edit_copy.png").Items.ToString = "Copy"
        .Add("List","fileview_text.png").Items.ToString = "List"
        .Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
    End With
    .EndUpdate()
End With
```

VB.NET for /COM

```
With AxRadialMenu1
    .BeginUpdate()
    .Expanded = True
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .set_HTMLPicture("arrow","arrow.png")
    .SubItemsBackColor = RGB(190,190,190)
    .ShadowColor = .SubItemsBackColor

.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialParentBorder,.SubIter



.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialItemsGridLines,-1)
    With .Items
        .Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
        .Add("Background-Color","color_fill.png").Items.ToString = "Background"
        .Add("Font","format_font_size_less.png").Items.ToString = "Font"
        .Add("Undo","edit_undo.png").Items.ToString = "Undo"
        .Add("Redo","edit_redo.png").Items.ToString = "Redo"
        .Add("Copy","edit_copy.png").Items.ToString = "Copy"
        .Add("List","fileview_text.png").Items.ToString = "List"
        .Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
```

```
    End With
    .EndUpdate()
End With
```

C++

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->PutPicturesPath(L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
spRadialMenu1->PutHTMLPicture(L"arrow","arrow.png");
spRadialMenu1->PutSubItemsBackColor(RGB(190,190,190));
spRadialMenu1->PutShadowColor(spRadialMenu1->GetSubItemsBackColor());
spRadialMenu1-
>PutRadialLineColor(EXRADIALMENULib::exRadialParentBorder,spRadialMenu1-
>GetSubItemsBackColor());
spRadialMenu1->PutRadialLineColor(EXRADIALMENULib::exRadialItemsGridLines,-1);
EXRADIALMENULib::IItemsPtr var_Items = spRadialMenu1->GetItems();
    var_Items->Add(L"Foreground-Color","color_line.png",vtMissing)->GetItems()-
>PutToString(L"Foreground");
    var_Items->Add(L"Background-Color","color_fill.png",vtMissing)->GetItems()-
>PutToString(L"Background");
    var_Items->Add(L"Font","format_font_size_less.png",vtMissing)->GetItems()-
>PutToString(L"Font");
    var_Items->Add(L"Undo","edit_undo.png",vtMissing)->GetItems()-
>PutToString(L"Undo");
    var_Items->Add(L"Redo","edit_redo.png",vtMissing)->GetItems()-
```

```
>PutToString(L"Redo");
   var_Items->Add(L"Copy","edit_copy.png",vtMissing)->GetItems()-
>PutToString(L"Copy");
   var_Items->Add(L"List","fileview_text.png",vtMissing)->GetItems()-
>PutToString(L"List");
   var_Items->Add(L"Tag","checkmark_korganizer.png",vtMissing)->GetItems()-
>PutToString(L"Tag");
spRadialMenu1->EndUpdate();
```

C++ Builder

```
RadialMenu1->BeginUpdate();
RadialMenu1->Expanded = true;
RadialMenu1->PicturesPath = L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
RadialMenu1->HTMLPicture[L"arrow"] = TVariant("arrow.png");
RadialMenu1->SubItemsBackColor = RGB(190,190,190);
RadialMenu1->ShadowColor = RadialMenu1->SubItemsBackColor;
RadialMenu1-
>RadialLineColor[ExradialmenulibÂ_tlb::RadialLineEnum::exRadialParentBorder] =
RadialMenu1->SubItemsBackColor;
RadialMenu1-
>RadialLineColor[ExradialmenulibÂ_tlb::RadialLineEnum::exRadialItemsGridLines] = -1;
ExradialmenulibÂ_tlb::IItemsPtr var_Items = RadialMenu1->Items;
   var_Items->Add(L"Foreground-Color",TVariant("color_line.png"),TNoParam())-
>Items->ToString = L"Foreground";
   var_Items->Add(L"Background-Color",TVariant("color_fill.png"),TNoParam())-
>Items->ToString = L"Background";
   var_Items->Add(L"Font",TVariant("format_font_size_less.png"),TNoParam())-
>Items->ToString = L"Font";
   var_Items->Add(L"Undo",TVariant("edit_undo.png"),TNoParam())->Items-
>ToString = L"Undo";
   var_Items->Add(L"Redo",TVariant("edit_redo.png"),TNoParam())->Items->ToString
= L"Redo";
   var_Items->Add(L"Copy",TVariant("edit_copy.png"),TNoParam())->Items-
>ToString = L"Copy";
```

```
   var_Items->Add(L"List",TVariant("fileview_text.png"),TNoParam())->Items->ToString = L"List";
   var_Items->Add(L"Tag",TVariant("checkmark_korganizer.png"),TNoParam())->Items->ToString = L"Tag";
RadialMenu1->EndUpdate();
```

C#

```
exradialmenu1.BeginUpdate();
exradialmenu1.Expanded = true;
exradialmenu1.PicturesPath = "C:\\Program Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
exradialmenu1.set_HTMLPicture("arrow","arrow.png");
exradialmenu1.SubItemsBackColor = Color.FromArgb(190,190,190);
exradialmenu1.ShadowColor = exradialmenu1.SubItemsBackColor;
exradialmenu1.set_RadialLineColor(exontrol.EXRADIALMENULib.RadialLineEnum.exRac

exradialmenu1.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exR

exontrol.EXRADIALMENULib.Items var_Items = exradialmenu1.Items;
   var_Items.Add("Foreground-Color","color_line.png",null).Items.ToString = "Foreground";
   var_Items.Add("Background-Color","color_fill.png",null).Items.ToString = "Background";
   var_Items.Add("Font","format_font_size_less.png",null).Items.ToString = "Font";
   var_Items.Add("Undo","edit_undo.png",null).Items.ToString = "Undo";
   var_Items.Add("Redo","edit_redo.png",null).Items.ToString = "Redo";
   var_Items.Add("Copy","edit_copy.png",null).Items.ToString = "Copy";
   var_Items.Add("List","fileview_text.png",null).Items.ToString = "List";
   var_Items.Add("Tag","checkmark_korganizer.png",null).Items.ToString = "Tag";
exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
```

```
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.Expanded = true;
   RadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
   RadialMenu1.HTMLPicture("arrow") = "arrow.png";
   RadialMenu1.SubItemsBackColor = 12500670;
   RadialMenu1.ShadowColor = RadialMenu1.SubItemsBackColor;
   RadialMenu1.RadialLineColor(6) = RadialMenu1.SubItemsBackColor;
   RadialMenu1.RadialLineColor(5) = -1;
   var var_Items = RadialMenu1.Items;
      var_Items.Add("Foreground-Color","color_line.png",null).Items.ToString =
"Foreground";
      var_Items.Add("Background-Color","color_fill.png",null).Items.ToString =
"Background";
      var_Items.Add("Font","format_font_size_less.png",null).Items.ToString = "Font";
      var_Items.Add("Undo","edit_undo.png",null).Items.ToString = "Undo";
      var_Items.Add("Redo","edit_redo.png",null).Items.ToString = "Redo";
      var_Items.Add("Copy","edit_copy.png",null).Items.ToString = "Copy";
      var_Items.Add("List","fileview_text.png",null).Items.ToString = "List";
      var_Items.Add("Tag","checkmark_korganizer.png",null).Items.ToString = "Tag";
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>
```

```
<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .Expanded = True
      .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
      .HTMLPicture("arrow") = "arrow.png"
      .SubItemsBackColor = RGB(190,190,190)
      .ShadowColor = .SubItemsBackColor
      .RadialLineColor(6) = .SubItemsBackColor
      .RadialLineColor(5) = -1
      With .Items
         .Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
         .Add("Background-Color","color_fill.png").Items.ToString = "Background"
         .Add("Font","format_font_size_less.png").Items.ToString = "Font"
         .Add("Undo","edit_undo.png").Items.ToString = "Undo"
         .Add("Redo","edit_redo.png").Items.ToString = "Redo"
         .Add("Copy","edit_copy.png").Items.ToString = "Copy"
         .Add("List","fileview_text.png").Items.ToString = "List"
         .Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
      End With
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.Expanded = true;
axRadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
axRadialMenu1.set_HTMLPicture("arrow","arrow.png");
axRadialMenu1.SubItemsBackColor = Color.FromArgb(190,190,190);
axRadialMenu1.ShadowColor = axRadialMenu1.SubItemsBackColor;
```

```
axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialParen

axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialItems

EXRADIALMENULib.Items var_Items = axRadialMenu1.Items;
   var_Items.Add("Foreground-Color","color_line.png",null).Items.ToString =
"Foreground";
   var_Items.Add("Background-Color","color_fill.png",null).Items.ToString =
"Background";
   var_Items.Add("Font","format_font_size_less.png",null).Items.ToString = "Font";
   var_Items.Add("Undo","edit_undo.png",null).Items.ToString = "Undo";
   var_Items.Add("Redo","edit_redo.png",null).Items.ToString = "Redo";
   var_Items.Add("Copy","edit_copy.png",null).Items.ToString = "Copy";
   var_Items.Add("List","fileview_text.png",null).Items.ToString = "List";
   var_Items.Add("Tag","checkmark_korganizer.png",null).Items.ToString = "Tag";
axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Item,com_Items,com_Items1;
   anytype var_Item,var_Items,var_Items1;
   ;

   super();

   exradialmenu1.BeginUpdate();
   exradialmenu1.Expanded(true);
   exradialmenu1.PicturesPath("C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
   exradialmenu1.HTMLPicture("arrow","arrow.png");
   exradialmenu1.SubItemsBackColor(WinApi::RGB2int(190,190,190));
   exradialmenu1.ShadowColor(exradialmenu1.SubItemsBackColor());

exradialmenu1.RadialLineColor(6/*exRadialParentBorder*/,exradialmenu1.SubItemsBac
```

```
   exradialmenu1.RadialLineColor(5/*exRadialItemsGridLines*/,-1);
   var_Items = exradialmenu1.Items(); com_Items = var_Items;
      var_Item = COM::createFromObject(com_Items.Add("Foreground-
Color","color_line.png")); com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
var_Items1;
      com_Items1.ToString("Foreground");
      var_Item = COM::createFromObject(com_Items.Add("Background-
Color","color_fill.png")); com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
var_Items1;
      com_Items1.ToString("Background");
      var_Item =
COM::createFromObject(com_Items.Add("Font","format_font_size_less.png"));
com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
var_Items1;
      com_Items1.ToString("Font");
      var_Item = COM::createFromObject(com_Items.Add("Undo","edit_undo.png"));
com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
var_Items1;
      com_Items1.ToString("Undo");
      var_Item = COM::createFromObject(com_Items.Add("Redo","edit_redo.png"));
com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
var_Items1;
      com_Items1.ToString("Redo");
      var_Item = COM::createFromObject(com_Items.Add("Copy","edit_copy.png"));
com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
var_Items1;
      com_Items1.ToString("Copy");
      var_Item = COM::createFromObject(com_Items.Add("List","fileview_text.png"));
com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
```

```
var_Items1;
      com_Items1.ToString("List");
      var_Item =
COM::createFromObject(com_Items.Add("Tag","checkmark_korganizer.png"));
com_Item = var_Item;
      var_Items1 = COM::createFromObject(com_Item).Items(); com_Items1 =
var_Items1;
      com_Items1.ToString("Tag");
   exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxRadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
   set_HTMLPicture('arrow','arrow.png');
   SubItemsBackColor := Color.FromArgb(190,190,190);
   ShadowColor := SubItemsBackColor;

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialParentBorder,SubItem



set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialItemsGridLines,$ffffff

   with Items do
   begin
      Add('Foreground-Color','color_line.png',Nil).Items.ToString := 'Foreground';
      Add('Background-Color','color_fill.png',Nil).Items.ToString := 'Background';
      Add('Font','format_font_size_less.png',Nil).Items.ToString := 'Font';
      Add('Undo','edit_undo.png',Nil).Items.ToString := 'Undo';
      Add('Redo','edit_redo.png',Nil).Items.ToString := 'Redo';
      Add('Copy','edit_copy.png',Nil).Items.ToString := 'Copy';
      Add('List','fileview_text.png',Nil).Items.ToString := 'List';
      Add('Tag','checkmark_korganizer.png',Nil).Items.ToString := 'Tag';
```

```
   end;
   EndUpdate();
end
```

Delphi (standard)

```
with RadialMenu1 do
begin
   BeginUpdate();
   Expanded := True;
   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
   HTMLPicture['arrow'] := 'arrow.png';
   SubItemsBackColor := RGB(190,190,190);
   ShadowColor := SubItemsBackColor;
   RadialLineColor[EXRADIALMENULib_TLB.exRadialParentBorder] :=
SubItemsBackColor;
   RadialLineColor[EXRADIALMENULib_TLB.exRadialItemsGridLines] := $ffffffff;
   with Items do
   begin
      Add('Foreground-Color','color_line.png',Null).Items.ToString := 'Foreground';
      Add('Background-Color','color_fill.png',Null).Items.ToString := 'Background';
      Add('Font','format_font_size_less.png',Null).Items.ToString := 'Font';
      Add('Undo','edit_undo.png',Null).Items.ToString := 'Undo';
      Add('Redo','edit_redo.png',Null).Items.ToString := 'Redo';
      Add('Copy','edit_copy.png',Null).Items.ToString := 'Copy';
      Add('List','fileview_text.png',Null).Items.ToString := 'List';
      Add('Tag','checkmark_korganizer.png',Null).Items.ToString := 'Tag';
   end;
   EndUpdate();
end
```

VFP

```
with thisform.RadialMenu1
   .BeginUpdate
   .Expanded = .T.
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .Object.HTMLPicture("arrow") = "arrow.png"
```

```
   .SubItemsBackColor = RGB(190,190,190)
   .ShadowColor = .SubItemsBackColor
   .Object.RadialLineColor(6) = .SubItemsBackColor
   .Object.RadialLineColor(5) = -1
   with .Items
      .Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
      .Add("Background-Color","color_fill.png").Items.ToString = "Background"
      .Add("Font","format_font_size_less.png").Items.ToString = "Font"
      .Add("Undo","edit_undo.png").Items.ToString = "Undo"
      .Add("Redo","edit_redo.png").Items.ToString = "Redo"
      .Add("Copy","edit_copy.png").Items.ToString = "Copy"
      .Add("List","fileview_text.png").Items.ToString = "List"
      .Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
local
oRadialMenu,var_Items,var_Items1,var_Items2,var_Items3,var_Items4,var_Items5,var_Ite



oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.Template = [HTMLPicture("arrow") = "arrow.png"] //
oRadialMenu.HTMLPicture("arrow") = "arrow.png"
oRadialMenu.SubItemsBackColor = 0xbebebe
oRadialMenu.ShadowColor = oRadialMenu.SubItemsBackColor
oRadialMenu.Template = [RadialLineColor(6) = SubItemsBackColor] //
oRadialMenu.RadialLineColor(6) = oRadialMenu.SubItemsBackColor
oRadialMenu.Template = [RadialLineColor(5) = -1] // oRadialMenu.RadialLineColor(5)
= -1
var_Items = oRadialMenu.Items
```

```
// var_Items.Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
var_Items1 = var_Items.Add("Foreground-Color","color_line.png").Items
with (oRadialMenu)
   TemplateDef = [dim var_Items1]
   TemplateDef = var_Items1
   Template = [var_Items1.ToString = "Foreground"]
endwith
// var_Items.Add("Background-Color","color_fill.png").Items.ToString = "Background"
var_Items2 = var_Items.Add("Background-Color","color_fill.png").Items
with (oRadialMenu)
   TemplateDef = [dim var_Items2]
   TemplateDef = var_Items2
   Template = [var_Items2.ToString = "Background"]
endwith
// var_Items.Add("Font","format_font_size_less.png").Items.ToString = "Font"
var_Items3 = var_Items.Add("Font","format_font_size_less.png").Items
with (oRadialMenu)
   TemplateDef = [dim var_Items3]
   TemplateDef = var_Items3
   Template = [var_Items3.ToString = "Font"]
endwith
// var_Items.Add("Undo","edit_undo.png").Items.ToString = "Undo"
var_Items4 = var_Items.Add("Undo","edit_undo.png").Items
with (oRadialMenu)
   TemplateDef = [dim var_Items4]
   TemplateDef = var_Items4
   Template = [var_Items4.ToString = "Undo"]
endwith
// var_Items.Add("Redo","edit_redo.png").Items.ToString = "Redo"
var_Items5 = var_Items.Add("Redo","edit_redo.png").Items
with (oRadialMenu)
   TemplateDef = [dim var_Items5]
   TemplateDef = var_Items5
   Template = [var_Items5.ToString = "Redo"]
endwith
// var_Items.Add("Copy","edit_copy.png").Items.ToString = "Copy"
var_Items6 = var_Items.Add("Copy","edit_copy.png").Items
```

```
   with (oRadialMenu)
      TemplateDef = [dim var_Items6]
      TemplateDef = var_Items6
      Template = [var_Items6.ToString = "Copy"]
   endwith
   // var_Items.Add("List","fileview_text.png").Items.ToString = "List"
   var_Items7 = var_Items.Add("List","fileview_text.png").Items
   with (oRadialMenu)
      TemplateDef = [dim var_Items7]
      TemplateDef = var_Items7
      Template = [var_Items7.ToString = "List"]
   endwith
   // var_Items.Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
   var_Items8 = var_Items.Add("Tag","checkmark_korganizer.png").Items
   with (oRadialMenu)
      TemplateDef = [dim var_Items8]
      TemplateDef = var_Items8
      Template = [var_Items8.ToString = "Tag"]
   endwith
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oRadialMenu as P
Dim var_Items as P
Dim var_Items1 as local
Dim var_Items2 as local
Dim var_Items3 as local
Dim var_Items4 as local
Dim var_Items5 as local
Dim var_Items6 as local
Dim var_Items7 as local
Dim var_Items8 as local

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
```

```
oRadialMenu.Expanded = .t.
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.Template = "HTMLPicture(`arrow`) = `arrow.png`" //
oRadialMenu.HTMLPicture("arrow") = "arrow.png"
oRadialMenu.SubItemsBackColor = 12500670
oRadialMenu.ShadowColor = oRadialMenu.SubItemsBackColor
oRadialMenu.Template = "RadialLineColor(6) = SubItemsBackColor" //
oRadialMenu.RadialLineColor(6) = oRadialMenu.SubItemsBackColor
oRadialMenu.Template = "RadialLineColor(5) = -1" //
oRadialMenu.RadialLineColor(5) = -1
var_Items = oRadialMenu.Items
   ' var_Items.Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
   var_Items1 = var_Items.Add("Foreground-Color","color_line.png").Items
   oRadialMenu.TemplateDef = "dim var_Items1"
   oRadialMenu.TemplateDef = var_Items1
   oRadialMenu.Template = "var_Items1.ToString = `Foreground`"

   ' var_Items.Add("Background-Color","color_fill.png").Items.ToString = "Background"
   var_Items2 = var_Items.Add("Background-Color","color_fill.png").Items
   oRadialMenu.TemplateDef = "dim var_Items2"
   oRadialMenu.TemplateDef = var_Items2
   oRadialMenu.Template = "var_Items2.ToString = `Background`"

   ' var_Items.Add("Font","format_font_size_less.png").Items.ToString = "Font"
   var_Items3 = var_Items.Add("Font","format_font_size_less.png").Items
   oRadialMenu.TemplateDef = "dim var_Items3"
   oRadialMenu.TemplateDef = var_Items3
   oRadialMenu.Template = "var_Items3.ToString = `Font`"

   ' var_Items.Add("Undo","edit_undo.png").Items.ToString = "Undo"
   var_Items4 = var_Items.Add("Undo","edit_undo.png").Items
   oRadialMenu.TemplateDef = "dim var_Items4"
   oRadialMenu.TemplateDef = var_Items4
   oRadialMenu.Template = "var_Items4.ToString = `Undo`"

   ' var_Items.Add("Redo","edit_redo.png").Items.ToString = "Redo"
```

```
    var_Items5 = var_Items.Add("Redo","edit_redo.png").Items
    oRadialMenu.TemplateDef = "dim var_Items5"
    oRadialMenu.TemplateDef = var_Items5
    oRadialMenu.Template = "var_Items5.ToString = `Redo`"

    ' var_Items.Add("Copy","edit_copy.png").Items.ToString = "Copy"
    var_Items6 = var_Items.Add("Copy","edit_copy.png").Items
    oRadialMenu.TemplateDef = "dim var_Items6"
    oRadialMenu.TemplateDef = var_Items6
    oRadialMenu.Template = "var_Items6.ToString = `Copy`"

    ' var_Items.Add("List","fileview_text.png").Items.ToString = "List"
    var_Items7 = var_Items.Add("List","fileview_text.png").Items
    oRadialMenu.TemplateDef = "dim var_Items7"
    oRadialMenu.TemplateDef = var_Items7
    oRadialMenu.Template = "var_Items7.ToString = `List`"

    ' var_Items.Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
    var_Items8 = var_Items.Add("Tag","checkmark_korganizer.png").Items
    oRadialMenu.TemplateDef = "dim var_Items8"
    oRadialMenu.TemplateDef = var_Items8
    oRadialMenu.Template = "var_Items8.ToString = `Tag`"

oRadialMenu.EndUpdate()
```

Visual Objects

```
local var_Items as IItems

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oDCOCX_Exontrol1:[HTMLPicture,"arrow"] := "arrow.png"
oDCOCX_Exontrol1:SubItemsBackColor := RGB(190,190,190)
oDCOCX_Exontrol1:ShadowColor := oDCOCX_Exontrol1:SubItemsBackColor
```

```
oDCOCX_Exontrol1:[RadialLineColor,exRadialParentBorder] :=
oDCOCX_Exontrol1:SubItemsBackColor
oDCOCX_Exontrol1:[RadialLineColor,exRadialItemsGridLines] := -1
var_Items := oDCOCX_Exontrol1:Items
   var_Items:Add("Foreground-Color","color_line.png",nil):Items:ToString :=
"Foreground"
   var_Items:Add("Background-Color","color_fill.png",nil):Items:ToString :=
"Background"
   var_Items:Add("Font","format_font_size_less.png",nil):Items:ToString := "Font"
   var_Items:Add("Undo","edit_undo.png",nil):Items:ToString := "Undo"
   var_Items:Add("Redo","edit_redo.png",nil):Items:ToString := "Redo"
   var_Items:Add("Copy","edit_copy.png",nil):Items:ToString := "Copy"
   var_Items:Add("List","fileview_text.png",nil):Items:ToString := "List"
   var_Items:Add("Tag","checkmark_korganizer.png",nil):Items:ToString := "Tag"
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oRadialMenu,var_Items

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.Expanded = true
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.HTMLPicture("arrow","arrow.png")
oRadialMenu.SubItemsBackColor = RGB(190,190,190)
oRadialMenu.ShadowColor = oRadialMenu.SubItemsBackColor
oRadialMenu.RadialLineColor(6,oRadialMenu.SubItemsBackColor)
oRadialMenu.RadialLineColor(5,-1)
var_Items = oRadialMenu.Items
   var_Items.Add("Foreground-Color","color_line.png").Items.ToString =
"Foreground"
   var_Items.Add("Background-Color","color_fill.png").Items.ToString = "Background"
   var_Items.Add("Font","format_font_size_less.png").Items.ToString = "Font"
   var_Items.Add("Undo","edit_undo.png").Items.ToString = "Undo"
```

```
    var_Items.Add("Redo","edit_redo.png").Items.ToString = "Redo"
    var_Items.Add("Copy","edit_copy.png").Items.ToString = "Copy"
    var_Items.Add("List","fileview_text.png").Items.ToString = "List"
    var_Items.Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
    Forward Send OnCreate
    Send ComBeginUpdate
    Set ComExpanded to True
    Set ComPicturesPath to "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    Set ComHTMLPicture "arrow" to "arrow.png"
    Set ComSubItemsBackColor to (RGB(190,190,190))
    Set ComShadowColor to (ComSubItemsBackColor(Self))
    Set ComRadialLineColor OLEexRadialParentBorder to
(ComSubItemsBackColor(Self))
    Set ComRadialLineColor OLEexRadialItemsGridLines to -1
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Variant voItem
        Get ComAdd of hoItems "Foreground-Color" "color_line.png" Nothing to
voItem
        Handle hoItem
        Get Create (RefClass(cComItem)) to hoItem
        Set pvComObject of hoItem to voItem
            Variant voItems1
            Get ComItems of hoItem to voItems1
            Handle hoItems1
            Get Create (RefClass(cComItems)) to hoItems1
            Set pvComObject of hoItems1 to voItems1
                Set ComToString of hoItems1 to "Foreground"
```

```
        Send Destroy to hoItems1
    Send Destroy to hoItem
    Variant voItem1
    Get ComAdd of hoItems "Background-Color" "color_fill.png" Nothing to
voItem1
    Handle hoItem1
    Get Create (RefClass(cComItem)) to hoItem1
    Set pvComObject of hoItem1 to voItem1
        Variant voItems2
        Get ComItems of hoItem1 to voItems2
        Handle hoItems2
        Get Create (RefClass(cComItems)) to hoItems2
        Set pvComObject of hoItems2 to voItems2
            Set ComToString of hoItems2 to "Background"
        Send Destroy to hoItems2
    Send Destroy to hoItem1
    Variant voItem2
    Get ComAdd of hoItems "Font" "format_font_size_less.png" Nothing to voItem2
    Handle hoItem2
    Get Create (RefClass(cComItem)) to hoItem2
    Set pvComObject of hoItem2 to voItem2
        Variant voItems3
        Get ComItems of hoItem2 to voItems3
        Handle hoItems3
        Get Create (RefClass(cComItems)) to hoItems3
        Set pvComObject of hoItems3 to voItems3
            Set ComToString of hoItems3 to "Font"
        Send Destroy to hoItems3
    Send Destroy to hoItem2
    Variant voItem3
    Get ComAdd of hoItems "Undo" "edit_undo.png" Nothing to voItem3
    Handle hoItem3
    Get Create (RefClass(cComItem)) to hoItem3
    Set pvComObject of hoItem3 to voItem3
        Variant voItems4
        Get ComItems of hoItem3 to voItems4
        Handle hoItems4
```

```
    Get Create (RefClass(cComItems)) to hoItems4
    Set pvComObject of hoItems4 to voItems4
        Set ComToString of hoItems4 to "Undo"
    Send Destroy to hoItems4
Send Destroy to hoItem3
Variant voItem4
Get ComAdd of hoItems "Redo" "edit_redo.png" Nothing to voItem4
Handle hoItem4
Get Create (RefClass(cComItem)) to hoItem4
Set pvComObject of hoItem4 to voItem4
    Variant voItems5
    Get ComItems of hoItem4 to voItems5
    Handle hoItems5
    Get Create (RefClass(cComItems)) to hoItems5
    Set pvComObject of hoItems5 to voItems5
        Set ComToString of hoItems5 to "Redo"
    Send Destroy to hoItems5
Send Destroy to hoItem4
Variant voItem5
Get ComAdd of hoItems "Copy" "edit_copy.png" Nothing to voItem5
Handle hoItem5
Get Create (RefClass(cComItem)) to hoItem5
Set pvComObject of hoItem5 to voItem5
    Variant voItems6
    Get ComItems of hoItem5 to voItems6
    Handle hoItems6
    Get Create (RefClass(cComItems)) to hoItems6
    Set pvComObject of hoItems6 to voItems6
        Set ComToString of hoItems6 to "Copy"
    Send Destroy to hoItems6
Send Destroy to hoItem5
Variant voItem6
Get ComAdd of hoItems "List" "fileview_text.png" Nothing to voItem6
Handle hoItem6
Get Create (RefClass(cComItem)) to hoItem6
Set pvComObject of hoItem6 to voItem6
    Variant voItems7
```

```
        Get ComItems of hoItem6 to voItems7
        Handle hoItems7
        Get Create (RefClass(cComItems)) to hoItems7
        Set pvComObject of hoItems7 to voItems7
          Set ComToString of hoItems7 to "List"
        Send Destroy to hoItems7
      Send Destroy to hoItem6
      Variant voItem7
      Get ComAdd of hoItems "Tag" "checkmark_korganizer.png" Nothing to voItem7
      Handle hoItem7
      Get Create (RefClass(cComItem)) to hoItem7
      Set pvComObject of hoItem7 to voItem7
        Variant voItems8
        Get ComItems of hoItem7 to voItems8
        Handle hoItems8
        Get Create (RefClass(cComItems)) to hoItems8
        Set pvComObject of hoItems8 to voItems8
          Set ComToString of hoItems8 to "Tag"
        Send Destroy to hoItems8
      Send Destroy to hoItem7
    Send Destroy to hoItems
    Send ComEndUpdate
End_Procedure
```

XBase++

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

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

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

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:BeginUpdate()
      oRadialMenu:Expanded := .T.
      oRadialMenu:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
      oRadialMenu:SetProperty("HTMLPicture","arrow","arrow.png")
      oRadialMenu:SetProperty("SubItemsBackColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 190,190,190 } )  , .F. ))
      oRadialMenu:SetProperty("ShadowColor",oRadialMenu:SubItemsBackColor())

oRadialMenu:SetProperty("RadialLineColor",6/*exRadialParentBorder*/,oRadialMenu:S

      oRadialMenu:SetProperty("RadialLineColor",5/*exRadialItemsGridLines*/,-1)
      oItems := oRadialMenu:Items()
         oItems:Add("Foreground-Color","color_line.png"):Items():ToString :=
"Foreground"
         oItems:Add("Background-Color","color_fill.png"):Items():ToString :=
"Background"
         oItems:Add("Font","format_font_size_less.png"):Items():ToString := "Font"
         oItems:Add("Undo","edit_undo.png"):Items():ToString := "Undo"
         oItems:Add("Redo","edit_redo.png"):Items():ToString := "Redo"
         oItems:Add("Copy","edit_copy.png"):Items():ToString := "Copy"
         oItems:Add("List","fileview_text.png"):Items():ToString := "List"
         oItems:Add("Tag","checkmark_korganizer.png"):Items():ToString := "Tag"
      oRadialMenu:EndUpdate()

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

```
    ENDDO
RETURN
```

property RadialMenu.PointerAngle as Double

Specifies the angle of the pointer to target another item or index.

Type	Description
Double	A double expression that specifies the angle of the pointer to target another item or index (in degrees).

By default, the PointerAngle property is 0. The PointerAngle property specifies the angle of the pointer to target another item or index. The PointerIndex property specifies the index within the radial menu to target the pointer. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The SelectedIndex property specifies the index of the item/slice to be selected.

The following screen show show a pointer over the control, with a different angle:

The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

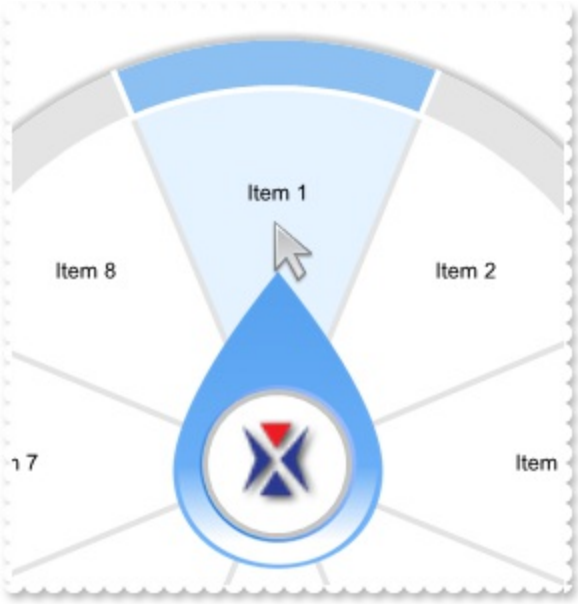
property RadialMenu.PointerIndex as Long

Specifies the index within the radial menu to target the pointer.

Type	Description
Long	A long expression that specifies the index within the radial menu to target the pointer.

By default, the PointerIndex property is 0. The PointerIndex property specifies the index within the radial menu to target the pointer. The PointerAngle property specifies the angle of the pointer to target another item or index. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The SelectedIndex property specifies the index of the item/slice to be selected.

The following screen show show a pointer over the control:



The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

The following sample shows how you can select an item, once the user clicks it:

VBA (MS Access, Excell...)

```
' SelectItem event - Notifies once the user selects an item.
Private Sub RadialMenu1_SelectItem(ByVal Item As Object)
    ' PointerIndex = Item.Index
    With RadialMenu1
        .SelectedIndex(3) = .PointerIndex
    End With
End Sub

With RadialMenu1
    .BeginUpdate
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .PointerPicture = "pointer.png"
    .SelBackAlpha(1) = 32
    .SelBackAlpha(2) = 128
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = False
    .SelForeColor(3) = RGB(0,0,0)
    .ParentSize = "36*dpi"
    .ParentImageHeight(-1) = "48*dpi"
    .ParentImageWidth(-1) = "48*dpi"
    .RadialLineSize(8) = -1
    .RadialLineAlpha(8) = 32
    .RadialLineColor(11) = -1
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .PointerIndex = 0
    .SelectedIndex(3) = .PointerIndex
    .EndUpdate
End With
```

VB6

```
' SelectItem event - Notifies once the user selects an item.
```

```
Private Sub RadialMenu1_SelectItem(ByVal Item As EXRADIALMENULibCtl.IItem)
    ' PointerIndex = Item.Index
    With RadialMenu1
        .SelectedIndex(exRadialFullItems) = .PointerIndex
    End With
End Sub

With RadialMenu1
    .BeginUpdate
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .PointerPicture = "pointer.png"
    .SelBackAlpha(exRadialItems) = 32
    .SelBackAlpha(exRadialSubItems) = 128
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = False
    .SelForeColor(exRadialFullItems) = RGB(0,0,0)
    .ParentSize = "36*dpi"
    .ParentImageHeight(exRadialMenuStateAll) = "48*dpi"
    .ParentImageWidth(exRadialMenuStateAll) = "48*dpi"
    .RadialLineSize(exRadialHotParent) = -1
    .RadialLineAlpha(exRadialHotParent) = 32
    .RadialLineColor(exRadialHotFullItem) = -1
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .PointerIndex = 0
    .SelectedIndex(exRadialFullItems) = .PointerIndex
    .EndUpdate
End With
```

VB.NET

```
' SelectItem event - Notifies once the user selects an item.
Private Sub Exradialmenu1_SelectItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXRADIALMENULib.Item) Handles Exradialmenu1.SelectItem
    ' PointerIndex = Item.Index
    With Exradialmenu1
```

```
.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.Po

   End With
End Sub

With Exradialmenu1
   .BeginUpdate()
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .PointerPicture = "pointer.png"
   .set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)

.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128

   .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
   .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
   .AllowHotPointer = False

.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,Colo

   .ParentSize = "36*dpi"

.set_ParentImageHeight(exontrol.EXRADIALMENULib.RadialMenuStateEnum.exRadialM


.set_ParentImageWidth(exontrol.EXRADIALMENULib.RadialMenuStateEnum.exRadialM


.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)


.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,3


.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotFullIte

   .Expanded = True
```

```
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .PointerIndex = 0

.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.Po

    .EndUpdate()
End With
```

VB.NET for /COM

```
' SelectItem event - Notifies once the user selects an item.
Private Sub AxRadialMenu1_SelectItem(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent) Handles
AxRadialMenu1.SelectItem
    ' PointerIndex = Item.Index
    With AxRadialMenu1

.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.PointerInd

    End With
End Sub

With AxRadialMenu1
    .BeginUpdate()
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .PointerPicture = "pointer.png"
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128)
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = False
    .set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)
    .ParentSize = "36*dpi"

.set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState
```

```
.set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState

   .set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)
   .set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32)
   .set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,-1)
   .Expanded = True
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
   .PointerIndex = 0

.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.PointerInd

   .EndUpdate()
End With
```

C++

```
// SelectItem event - Notifies once the user selects an item.
void OnSelectItemRadialMenu1(LPDISPATCH   Item)
{
   // PointerIndex = Item.Index
   /*
        Copy and paste the following directives to your header file as
        it defines the namespace 'EXRADIALMENULib' for the library: 'ExRadialMenu 1.0
Control Library'
        #import <ExRadialMenu.dll>
        using namespace EXRADIALMENULib;
    */
   EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
   spRadialMenu1-
>PutSelectedIndex(EXRADIALMENULib::exRadialFullItems,spRadialMenu1-
>GetPointerIndex());
}

EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
```

```
spRadialMenu1->PutPicturesPath(L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
spRadialMenu1->PutPointerPicture("pointer.png");
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialItems,32);
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialSubItems,128);
spRadialMenu1->PutPointerPictureY(L"y + (height-pheight)/2- 21*dpi");
spRadialMenu1->PutPointerPictureX(L"x + (width-pwidth)/2 + 1 * dpi");
spRadialMenu1->PutAllowHotPointer(VARIANT_FALSE);
spRadialMenu1->PutSelForeColor(EXRADIALMENULib::exRadialFullItems,RGB(0,0,0));
spRadialMenu1->PutParentSize(L"36*dpi");
spRadialMenu1-
>PutParentImageHeight(EXRADIALMENULib::exRadialMenuStateAll,L"48*dpi");
spRadialMenu1-
>PutParentImageWidth(EXRADIALMENULib::exRadialMenuStateAll,L"48*dpi");
spRadialMenu1->PutRadialLineSize(EXRADIALMENULib::exRadialHotParent,-1);
spRadialMenu1->PutRadialLineAlpha(EXRADIALMENULib::exRadialHotParent,32);
spRadialMenu1->PutRadialLineColor(EXRADIALMENULib::exRadialHotFullItem,-1);
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->GetItems()->PutToString(L"Item 1,Item 2,Item 3,Item 4,Item 5,Item
6,Item 7,Item 8");
spRadialMenu1->PutPointerIndex(0);
spRadialMenu1-
>PutSelectedIndex(EXRADIALMENULib::exRadialFullItems,spRadialMenu1-
>GetPointerIndex());
spRadialMenu1->EndUpdate();
```

C++ Builder

```
// SelectItem event - Notifies once the user selects an item.
void __fastcall TForm1::RadialMenu1SelectItem(TObject
*Sender,Exradialmenulib_tlb::IItem  *Item)
{
  // PointerIndex = Item.Index
  RadialMenu1-
>SelectedIndex[Exradialmenulib_tlb::RadialItemsEnum::exRadialFullItems] =
RadialMenu1->PointerIndex;
```

```
}

RadialMenu1->BeginUpdate();
RadialMenu1->PicturesPath = L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
RadialMenu1->set_PointerPicture(TVariant("pointer.png"));
RadialMenu1->SelBackAlpha[ExradialmenulibTlb::RadialItemsEnum::exRadialItems] =
32;
RadialMenu1-
>SelBackAlpha[ExradialmenulibTlb::RadialItemsEnum::exRadialSubItems] = 128;
RadialMenu1->PointerPictureY = L"y + (height-pheight)/2- 21*dpi";
RadialMenu1->PointerPictureX = L"x + (width-pwidth)/2 + 1 * dpi";
RadialMenu1->AllowHotPointer = false;
RadialMenu1-
>SelForeColor[ExradialmenulibTlb::RadialItemsEnum::exRadialFullItems] =
RGB(0,0,0);
RadialMenu1->ParentSize = L"36*dpi";
RadialMenu1-
>ParentImageHeight[ExradialmenulibTlb::RadialMenuStateEnum::exRadialMenuStateA
 = L"48*dpi";
RadialMenu1-
>ParentImageWidth[ExradialmenulibTlb::RadialMenuStateEnum::exRadialMenuStateAl
 = L"48*dpi";
RadialMenu1-
>RadialLineSize[ExradialmenulibTlb::RadialLineEnum::exRadialHotParent] = -1;
RadialMenu1-
>RadialLineAlpha[ExradialmenulibTlb::RadialLineEnum::exRadialHotParent] = 32;
RadialMenu1-
>RadialLineColor[ExradialmenulibTlb::RadialLineEnum::exRadialHotFullItem] = -1;
RadialMenu1->Expanded = true;
RadialMenu1->Items->ToString = L"Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
RadialMenu1->PointerIndex = 0;
RadialMenu1-
>SelectedIndex[ExradialmenulibTlb::RadialItemsEnum::exRadialFullItems] =
RadialMenu1->PointerIndex;
RadialMenu1->EndUpdate();
```

C#

```
// SelectItem event - Notifies once the user selects an item.
private void exradialmenu1_SelectItem(object
sender,exontrol.EXRADIALMENULib.Item   Item)
{
   // PointerIndex = Item.Index

exradialmenu1.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRad

}
//this.exradialmenu1.SelectItem += new
exontrol.EXRADIALMENULib.exg2antt.SelectItemEventHandler(this.exradialmenu1_Selec



exradialmenu1.BeginUpdate();
exradialmenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
exradialmenu1.PointerPicture = "pointer.png";
exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
exradialmenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
exradialmenu1.AllowHotPointer = false;
exradialmenu1.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadia

exradialmenu1.ParentSize = "36*dpi";
exradialmenu1.set_ParentImageHeight(exontrol.EXRADIALMENULib.RadialMenuStateE

exradialmenu1.set_ParentImageWidth(exontrol.EXRADIALMENULib.RadialMenuStateEr

exradialmenu1.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadia
```

```
exradialmenu1.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRad

exradialmenu1.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exR

exradialmenu1.Expanded = true;
exradialmenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8";
exradialmenu1.PointerIndex = 0;
exradialmenu1.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRad

exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<SCRIPT FOR="RadialMenu1" EVENT="SelectItem(Item)" LANGUAGE="JScript">
   // PointerIndex = Item.Index
   RadialMenu1.SelectedIndex(3) = RadialMenu1.PointerIndex;
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
   RadialMenu1.PointerPicture = "pointer.png";
   RadialMenu1.SelBackAlpha(1) = 32;
   RadialMenu1.SelBackAlpha(2) = 128;
   RadialMenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
   RadialMenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
   RadialMenu1.AllowHotPointer = false;
   RadialMenu1.SelForeColor(3) = 0;
```

```
   RadialMenu1.ParentSize = "36*dpi";
   RadialMenu1.ParentImageHeight(-1) = "48*dpi";
   RadialMenu1.ParentImageWidth(-1) = "48*dpi";
   RadialMenu1.RadialLineSize(8) = -1;
   RadialMenu1.RadialLineAlpha(8) = 32;
   RadialMenu1.RadialLineColor(11) = -1;
   RadialMenu1.Expanded = true;
   RadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
   RadialMenu1.PointerIndex = 0;
   RadialMenu1.SelectedIndex(3) = RadialMenu1.PointerIndex;
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<SCRIPT LANGUAGE="VBScript">
Function RadialMenu1_SelectItem(Item)
   ' PointerIndex = Item.Index
   With RadialMenu1
      .SelectedIndex(3) = .PointerIndex
   End With
End Function
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
```

```
      .PointerPicture = "pointer.png"
      .SelBackAlpha(1) = 32
      .SelBackAlpha(2) = 128
      .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
      .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
      .AllowHotPointer = False
      .SelForeColor(3) = RGB(0,0,0)
      .ParentSize = "36*dpi"
      .ParentImageHeight(-1) = "48*dpi"
      .ParentImageWidth(-1) = "48*dpi"
      .RadialLineSize(8) = -1
      .RadialLineAlpha(8) = 32
      .RadialLineColor(11) = -1
      .Expanded = True
      .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
      .PointerIndex = 0
      .SelectedIndex(3) = .PointerIndex
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
// SelectItem event - Notifies once the user selects an item.
private void axRadialMenu1_SelectItem(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent e)
{
   // PointerIndex = Item.Index

axRadialMenu1.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullIte

}
//this.axRadialMenu1.SelectItem += new
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEventHandler(this.axRadialMenu1_
```

```
axRadialMenu1.BeginUpdate();
axRadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
axRadialMenu1.PointerPicture = "pointer.png";
axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,3

axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubIte

axRadialMenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
axRadialMenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
axRadialMenu1.AllowHotPointer = false;
axRadialMenu1.set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullIter
(uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0)));
axRadialMenu1.ParentSize = "36*dpi";
axRadialMenu1.set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exR

axRadialMenu1.set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRa

axRadialMenu1.set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotPar

axRadialMenu1.set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotP

axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFu

axRadialMenu1.Expanded = true;
axRadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
axRadialMenu1.PointerIndex = 0;
axRadialMenu1.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullIte

axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
// SelectItem event - Notifies once the user selects an item.
void onEvent_SelectItem(COM   _Item)
{
   // PointerIndex = Item.Index
   ;

exradialmenu1.SelectedIndex(3/*exRadialFullItems*/,exradialmenu1.PointerIndex());
}

public void init()
{
   ;

   super();

   exradialmenu1.BeginUpdate();
   exradialmenu1.PicturesPath("C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
   exradialmenu1.PointerPicture("pointer.png");
   exradialmenu1.SelBackAlpha(1/*exRadialItems*/,32);
   exradialmenu1.SelBackAlpha(2/*exRadialSubItems*/,128);
   exradialmenu1.PointerPictureY("y + (height-pheight)/2- 21*dpi");
   exradialmenu1.PointerPictureX("x + (width-pwidth)/2 + 1 * dpi");
   exradialmenu1.AllowHotPointer(false);
   exradialmenu1.SelForeColor(3/*exRadialFullItems*/,WinApi::RGB2int(0,0,0));
   exradialmenu1.ParentSize("36*dpi");
   exradialmenu1.ParentImageHeight(-1/*exRadialMenuStateAll*/,"48*dpi");
   exradialmenu1.ParentImageWidth(-1/*exRadialMenuStateAll*/,"48*dpi");
   exradialmenu1.RadialLineSize(8/*exRadialHotParent*/,-1);
   exradialmenu1.RadialLineAlpha(8/*exRadialHotParent*/,32);
   exradialmenu1.RadialLineColor(11/*exRadialHotFullItem*/,-1);
   exradialmenu1.Expanded(true);
   exradialmenu1.Items().ToString("Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8");
   exradialmenu1.PointerIndex(0);

exradialmenu1.SelectedIndex(3/*exRadialFullItems*/,exradialmenu1.PointerIndex());
```

```
    exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
// SelectItem event - Notifies once the user selects an item.
procedure TWinForm1.AxRadialMenu1_SelectItem(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent);
begin
   // PointerIndex = Item.Index
   with AxRadialMenu1 do
   begin

set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,PointerInde

   end
end;

with AxRadialMenu1 do
begin
   BeginUpdate();
   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
   PointerPicture := 'pointer.png';
   set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32);
   set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128);
   PointerPictureY := 'y + (height-pheight)/2- 21*dpi';
   PointerPictureX := 'x + (width-pwidth)/2 + 1 * dpi';
   AllowHotPointer := False;
   set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,$0);
   ParentSize := '36*dpi';

set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState



set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuStateA

   set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1);
```

```
   set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32);

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,$ffffffff);

   Expanded := True;
   Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
   PointerIndex := 0;

set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,PointerInde

   EndUpdate();
end
```

Delphi (standard)

```
// SelectItem event - Notifies once the user selects an item.
procedure TForm1.RadialMenu1SelectItem(ASender: TObject; Item : IItem);
begin
   // PointerIndex = Item.Index
   with RadialMenu1 do
   begin
      SelectedIndex[EXRADIALMENULib_TLB.exRadialFullItems] := PointerIndex;
   end
end;

with RadialMenu1 do
begin
   BeginUpdate();
   PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
   PointerPicture := 'pointer.png';
   SelBackAlpha[EXRADIALMENULib_TLB.exRadialItems] := 32;
   SelBackAlpha[EXRADIALMENULib_TLB.exRadialSubItems] := 128;
   PointerPictureY := 'y + (height-pheight)/2- 21*dpi';
   PointerPictureX := 'x + (width-pwidth)/2 + 1 * dpi';
   AllowHotPointer := False;
   SelForeColor[EXRADIALMENULib_TLB.exRadialFullItems] := $0;
   ParentSize := '36*dpi';
```

```
  ParentImageHeight[EXRADIALMENULib_TLB.exRadialMenuStateAll] := '48*dpi';
  ParentImageWidth[EXRADIALMENULib_TLB.exRadialMenuStateAll] := '48*dpi';
  RadialLineSize[EXRADIALMENULib_TLB.exRadialHotParent] := -1;
  RadialLineAlpha[EXRADIALMENULib_TLB.exRadialHotParent] := 32;
  RadialLineColor[EXRADIALMENULib_TLB.exRadialHotFullItem] := $ffffffff;
  Expanded := True;
  Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
  PointerIndex := 0;
  SelectedIndex[EXRADIALMENULib_TLB.exRadialFullItems] := PointerIndex;
  EndUpdate();
end
```

VFP

```
*** SelectItem event - Notifies once the user selects an item. ***
LPARAMETERS Item
   *** PointerIndex = Item.Index
   with thisform.RadialMenu1
      .Object.SelectedIndex(3) = .PointerIndex
   endwith

with thisform.RadialMenu1
   .BeginUpdate
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .PointerPicture = "pointer.png"
   .Object.SelBackAlpha(1) = 32
   .Object.SelBackAlpha(2) = 128
   .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
   .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
   .AllowHotPointer = .F.
   .Object.SelForeColor(3) = RGB(0,0,0)
   .ParentSize = "36*dpi"
   .Object.ParentImageHeight(-1) = "48*dpi"
   .Object.ParentImageWidth(-1) = "48*dpi"
   .Object.RadialLineSize(8) = -1
   .Object.RadialLineAlpha(8) = 32
   .Object.RadialLineColor(11) = -1
```

```
    .Expanded = .T.
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .PointerIndex = 0
    .Object.SelectedIndex(3) = .PointerIndex
    .EndUpdate
endwith
```

dBASE Plus

```
/*
with (this.EXRADIALMENUACTIVEXCONTROL1.nativeObject)
    SelectItem = class::nativeObject_SelectItem
endwith
*/
// Notifies once the user selects an item.
function nativeObject_SelectItem(Item)
   /* PointerIndex = Item.Index */
   oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
   oRadialMenu.Template = [SelectedIndex(3) = PointerIndex] //
oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
return

local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.Template = [SelBackAlpha(1) = 32] // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = [SelBackAlpha(2) = 128] // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = false
oRadialMenu.Template = [SelForeColor(3) = 0] // oRadialMenu.SelForeColor(3) = 0x0
```

```
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.Template = [ParentImageHeight(-1) = "48*dpi"] //
oRadialMenu.ParentImageHeight(-1) = "48*dpi"
oRadialMenu.Template = [ParentImageWidth(-1) = "48*dpi"] //
oRadialMenu.ParentImageWidth(-1) = "48*dpi"
oRadialMenu.Template = [RadialLineSize(8) = -1] // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = [RadialLineAlpha(8) = 32] //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = [RadialLineColor(11) = -1] //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.PointerIndex = 0
oRadialMenu.Template = [SelectedIndex(3) = PointerIndex] //
oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
' Notifies once the user selects an item.
function SelectItem as v (Item  as  OLE::Exontrol.RadialMenu.1::IItem)
   ' PointerIndex = Item.Index
   oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
   oRadialMenu.Template = "SelectedIndex(3) = PointerIndex" //
oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
end function

Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
```

```
oRadialMenu.Template = "SelBackAlpha(1) = 32" // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = "SelBackAlpha(2) = 128" // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = .f.
oRadialMenu.Template = "SelForeColor(3) = 0" // oRadialMenu.SelForeColor(3) = 0
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.Template = "ParentImageHeight(-1) = `48*dpi`" //
oRadialMenu.ParentImageHeight(-1) = "48*dpi"
oRadialMenu.Template = "ParentImageWidth(-1) = `48*dpi`" //
oRadialMenu.ParentImageWidth(-1) = "48*dpi"
oRadialMenu.Template = "RadialLineSize(8) = -1" // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = "RadialLineAlpha(8) = 32" //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = "RadialLineColor(11) = -1" //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = .t.
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.PointerIndex = 0
oRadialMenu.Template = "SelectedIndex(3) = PointerIndex" //
oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
oRadialMenu.EndUpdate()
```

Visual Objects

```
METHOD OCX_Exontrol1SelectItem(Item) CLASS MainDialog
   // SelectItem event - Notifies once the user selects an item.
   // PointerIndex = Item.Index
   oDCOCX_Exontrol1:[SelectedIndex,exRadialFullItems] :=
oDCOCX_Exontrol1:PointerIndex
RETURN NIL
```

```
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oDCOCX_Exontrol1:PointerPicture := "pointer.png"
oDCOCX_Exontrol1:[SelBackAlpha,exRadialItems] := 32
oDCOCX_Exontrol1:[SelBackAlpha,exRadialSubItems] := 128
oDCOCX_Exontrol1:PointerPictureY := "y + (height-pheight)/2- 21*dpi"
oDCOCX_Exontrol1:PointerPictureX := "x + (width-pwidth)/2 + 1 * dpi"
oDCOCX_Exontrol1:AllowHotPointer := false
oDCOCX_Exontrol1:[SelForeColor,exRadialFullItems] := RGB(0,0,0)
oDCOCX_Exontrol1:ParentSize := "36*dpi"
oDCOCX_Exontrol1:[ParentImageHeight,exRadialMenuStateAll] := "48*dpi"
oDCOCX_Exontrol1:[ParentImageWidth,exRadialMenuStateAll] := "48*dpi"
oDCOCX_Exontrol1:[RadialLineSize,exRadialHotParent] := -1
oDCOCX_Exontrol1:[RadialLineAlpha,exRadialHotParent] := 32
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotFullItem] := -1
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:Items:ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
oDCOCX_Exontrol1:PointerIndex := 0
oDCOCX_Exontrol1:[SelectedIndex,exRadialFullItems] :=
oDCOCX_Exontrol1:PointerIndex
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
/*begin event SelectItem(oleobject  Item) - Notifies once the user selects an item.*/
/*
   PointerIndex = Item.Index
   oRadialMenu = ole_1.Object
   oRadialMenu.SelectedIndex(3,oRadialMenu.PointerIndex)
*/
/*end event SelectItem*/

OleObject oRadialMenu
```

```
oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.SelBackAlpha(1,32)
oRadialMenu.SelBackAlpha(2,128)
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = false
oRadialMenu.SelForeColor(3,RGB(0,0,0))
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.ParentImageHeight(-1,"48*dpi")
oRadialMenu.ParentImageWidth(-1,"48*dpi")
oRadialMenu.RadialLineSize(8,-1)
oRadialMenu.RadialLineAlpha(8,32)
oRadialMenu.RadialLineColor(11,-1)
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.PointerIndex = 0
oRadialMenu.SelectedIndex(3,oRadialMenu.PointerIndex)
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
// Notifies once the user selects an item.
Procedure OnComSelectItem Variant   llItem
   Forward Send OnComSelectItem llItem
   // PointerIndex = Item.Index
   Set ComSelectedIndex OLEexRadialFullItems to (ComPointerIndex(Self))
End_Procedure

Procedure OnCreate
   Forward Send OnCreate
```

```
    Send ComBeginUpdate
    Set ComPicturesPath to "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    Set ComPointerPicture to "pointer.png"
    Set ComSelBackAlpha OLEexRadialItems to 32
    Set ComSelBackAlpha OLEexRadialSubItems to 128
    Set ComPointerPictureY to "y + (height-pheight)/2- 21*dpi"
    Set ComPointerPictureX to "x + (width-pwidth)/2 + 1 * dpi"
    Set ComAllowHotPointer to False
    Set ComSelForeColor OLEexRadialFullItems to (RGB(0,0,0))
    Set ComParentSize to "36*dpi"
    Set ComParentImageHeight OLEexRadialMenuStateAll to "48*dpi"
    Set ComParentImageWidth OLEexRadialMenuStateAll to "48*dpi"
    Set ComRadialLineSize OLEexRadialHotParent to -1
    Set ComRadialLineAlpha OLEexRadialHotParent to 32
    Set ComRadialLineColor OLEexRadialHotFullItem to -1
    Set ComExpanded to True
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Set ComToString of hoItems to "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
    Send Destroy to hoItems
    Set ComPointerIndex to 0
    Set ComSelectedIndex OLEexRadialFullItems to (ComPointerIndex(Self))
    Send ComEndUpdate
End_Procedure
```

XBase++

```
PROCEDURE OnSelectItem(oRadialMenu,Item)
   /*PointerIndex = Item.Index*/

oRadialMenu:SetProperty("SelectedIndex",3/*exRadialFullItems*/,oRadialMenu:Pointe

RETURN
```

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

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

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:SelectItem := {|Item| OnSelectItem(oRadialMenu,Item)} /*Notifies
once the user selects an item.*/

      oRadialMenu:BeginUpdate()
      oRadialMenu:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
      oRadialMenu:PointerPicture := "pointer.png"
      oRadialMenu:SetProperty("SelBackAlpha",1/*exRadialItems*/,32)
      oRadialMenu:SetProperty("SelBackAlpha",2/*exRadialSubItems*/,128)
      oRadialMenu:PointerPictureY := "y + (height-pheight)/2- 21*dpi"
      oRadialMenu:PointerPictureX := "x + (width-pwidth)/2 + 1 * dpi"
      oRadialMenu:AllowHotPointer := .F.

oRadialMenu:SetProperty("SelForeColor",3/*exRadialFullItems*/,AutomationTranslateC
 GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oRadialMenu:ParentSize := "36*dpi"

oRadialMenu:SetProperty("ParentImageHeight",-1/*exRadialMenuStateAll*/,"48*dpi")
```

```
oRadialMenu:SetProperty("ParentImageWidth",-1/*exRadialMenuStateAll*/,"48*dpi")
      oRadialMenu:SetProperty("RadialLineSize",8/*exRadialHotParent*/,-1)
      oRadialMenu:SetProperty("RadialLineAlpha",8/*exRadialHotParent*/,32)
      oRadialMenu:SetProperty("RadialLineColor",11/*exRadialHotFullItem*/,-1)
      oRadialMenu:Expanded := .T.
      oRadialMenu:Items():ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
      oRadialMenu:PointerIndex := 0

oRadialMenu:SetProperty("SelectedIndex",3/*exRadialFullItems*/,oRadialMenu:Pointe

      oRadialMenu:EndUpdate()

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

property RadialMenu.PointerPicture as Variant

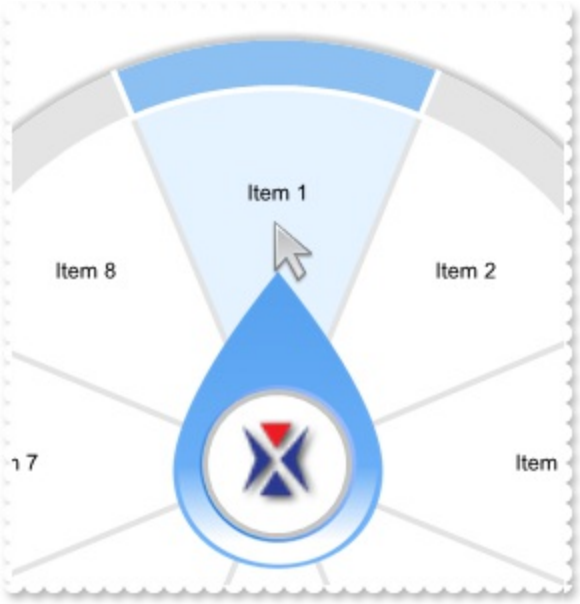
Indicates the picture to be shown on the pointer zone's background.

Type	Description

Variant	A VARIANT expression that indicates the picture to be shown on the pointer zone's background. It can be one of the following:
	
	• A String expression indicates:
	  ◦ a name of a picture file in the PicturePath folder. For instance, PointerPicture = "favorites.png", loads the favorites.png file if found in the PicturePath folder.
	  ◦ a picture file including its absolute path. For instance, PointerPicture = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor loads the favorites.png file from absolute path
	  ◦ a key of the HTML picture, previously loaded by the HTMLPicture method. For instance, PointerPicture = "pic1", loads the HTML picture with the key pic1, so the pic1 should be load previously with a HTMLPicture call like HTMLPicture("pic1") = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images\favor
	  ◦ an encode BASE64 string of a picture file. The Exontrol's ExImages Tool encode/decode BASE64 strings from/to pictures. In this case, the string starts with "gB..", "gC.." and so on.
	• A Picture object that indicates the picture to be displayed. For instance, PointerPicture = LoadPicture("picture.jpg")
	
	If no picture/image is found, the item displays no picture/image.

By default, the PointerPicture property is empty. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The SelectedIndex property specifies the index of the item/slice to be selected.

The following screen show show a pointer over the control:



The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

The PointerIndex property specifies the index within the radial menu to target the pointer.
The PointerAngle property specifies the angle of the pointer to target another item or index.

The following sample shows how you can select an item, once the user clicks it:

VBA (MS Access, Excell...)

```
' SelectItem event - Notifies once the user selects an item.
Private Sub RadialMenu1_SelectItem(ByVal Item As Object)
    ' PointerIndex = Item.Index
    With RadialMenu1
        .SelectedIndex(3) = .PointerIndex
    End With
End Sub
```

```
With RadialMenu1
   .BeginUpdate
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .PointerPicture = "pointer.png"
   .SelBackAlpha(1) = 32
   .SelBackAlpha(2) = 128
   .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
   .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
   .AllowHotPointer = False
   .SelForeColor(3) = RGB(0,0,0)
   .ParentSize = "36*dpi"
   .ParentImageHeight(-1) = "48*dpi"
   .ParentImageWidth(-1) = "48*dpi"
   .RadialLineSize(8) = -1
   .RadialLineAlpha(8) = 32
   .RadialLineColor(11) = -1
   .Expanded = True
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
   .PointerIndex = 0
   .SelectedIndex(3) = .PointerIndex
   .EndUpdate
End With
```

VB6

```
' SelectItem event - Notifies once the user selects an item.
Private Sub RadialMenu1_SelectItem(ByVal Item As EXRADIALMENULibCtl.IItem)
   ' PointerIndex = Item.Index
   With RadialMenu1
      .SelectedIndex(exRadialFullItems) = .PointerIndex
   End With
End Sub

With RadialMenu1
   .BeginUpdate
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .PointerPicture = "pointer.png"
```

```
    .SelBackAlpha(exRadialItems) = 32
    .SelBackAlpha(exRadialSubItems) = 128
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = False
    .SelForeColor(exRadialFullItems) = RGB(0,0,0)
    .ParentSize = "36*dpi"
    .ParentImageHeight(exRadialMenuStateAll) = "48*dpi"
    .ParentImageWidth(exRadialMenuStateAll) = "48*dpi"
    .RadialLineSize(exRadialHotParent) = -1
    .RadialLineAlpha(exRadialHotParent) = 32
    .RadialLineColor(exRadialHotFullItem) = -1
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .PointerIndex = 0
    .SelectedIndex(exRadialFullItems) = .PointerIndex
    .EndUpdate
End With
```

VB.NET

```
' SelectItem event - Notifies once the user selects an item.
Private Sub Exradialmenu1_SelectItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXRADIALMENULib.Item) Handles Exradialmenu1.SelectItem
    ' PointerIndex = Item.Index
    With Exradialmenu1

.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.Po

    End With
End Sub

With Exradialmenu1
    .BeginUpdate()
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .PointerPicture = "pointer.png"
    .set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)
```

```
.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128

   .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
   .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
   .AllowHotPointer = False

.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,Colo

   .ParentSize = "36*dpi"

.set_ParentImageHeight(exontrol.EXRADIALMENULib.RadialMenuStateEnum.exRadialM


.set_ParentImageWidth(exontrol.EXRADIALMENULib.RadialMenuStateEnum.exRadialM


.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)


.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,3


.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotFullIte

   .Expanded = True
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
   .PointerIndex = 0

.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.Po

   .EndUpdate()
End With
```

VB.NET for /COM

```
' SelectItem event - Notifies once the user selects an item.
```

```
Private Sub AxRadialMenu1_SelectItem(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent) Handles
AxRadialMenu1.SelectItem
    ' PointerIndex = Item.Index
    With AxRadialMenu1

.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.PointerInd

    End With
End Sub

With AxRadialMenu1
    .BeginUpdate()
    .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
    .PointerPicture = "pointer.png"
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128)
    .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
    .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
    .AllowHotPointer = False
    .set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)
    .ParentSize = "36*dpi"

.set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState



.set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState

    .set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)
    .set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32)
    .set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,-1)
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .PointerIndex = 0

.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,.PointerInd
```

```
    .EndUpdate()
End With
```

C++

```
// SelectItem event - Notifies once the user selects an item.
void OnSelectItemRadialMenu1(LPDISPATCH  Item)
{
    // PointerIndex = Item.Index
    /*
        Copy and paste the following directives to your header file as
        it defines the namespace 'EXRADIALMENULib' for the library: 'ExRadialMenu 1.0
Control Library'
        #import <ExRadialMenu.dll>
        using namespace EXRADIALMENULib;
     */
    EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
    spRadialMenu1-
>PutSelectedIndex(EXRADIALMENULib::exRadialFullItems,spRadialMenu1-
>GetPointerIndex());
}

EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutPicturesPath(L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
spRadialMenu1->PutPointerPicture("pointer.png");
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialItems,32);
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialSubItems,128);
spRadialMenu1->PutPointerPictureY(L"y + (height-pheight)/2- 21*dpi");
spRadialMenu1->PutPointerPictureX(L"x + (width-pwidth)/2 + 1 * dpi");
spRadialMenu1->PutAllowHotPointer(VARIANT_FALSE);
spRadialMenu1->PutSelForeColor(EXRADIALMENULib::exRadialFullItems,RGB(0,0,0));
spRadialMenu1->PutParentSize(L"36*dpi");
spRadialMenu1-
```

```
>PutParentImageHeight(EXRADIALMENULib::exRadialMenuStateAll,L"48*dpi");
spRadialMenu1-
>PutParentImageWidth(EXRADIALMENULib::exRadialMenuStateAll,L"48*dpi");
spRadialMenu1->PutRadialLineSize(EXRADIALMENULib::exRadialHotParent,-1);
spRadialMenu1->PutRadialLineAlpha(EXRADIALMENULib::exRadialHotParent,32);
spRadialMenu1->PutRadialLineColor(EXRADIALMENULib::exRadialHotFullItem,-1);
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->GetItems()->PutToString(L"Item 1,Item 2,Item 3,Item 4,Item 5,Item
6,Item 7,Item 8");
spRadialMenu1->PutPointerIndex(0);
spRadialMenu1-
>PutSelectedIndex(EXRADIALMENULib::exRadialFullItems,spRadialMenu1-
>GetPointerIndex());
spRadialMenu1->EndUpdate();
```

C++ Builder

```
// SelectItem event - Notifies once the user selects an item.
void __fastcall TForm1::RadialMenu1SelectItem(TObject
*Sender,Exradialmenulib_tlb::IItem  *Item)
{
   // PointerIndex = Item.Index
   RadialMenu1-
>SelectedIndex[Exradialmenulib_tlb::RadialItemsEnum::exRadialFullItems] =
RadialMenu1->PointerIndex;
}

RadialMenu1->BeginUpdate();
RadialMenu1->PicturesPath = L"C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
RadialMenu1->set_PointerPicture(TVariant("pointer.png"));
RadialMenu1->SelBackAlpha[Exradialmenulib_tlb::RadialItemsEnum::exRadialItems] =
32;
RadialMenu1-
>SelBackAlpha[Exradialmenulib_tlb::RadialItemsEnum::exRadialSubItems] = 128;
RadialMenu1->PointerPictureY = L"y + (height-pheight)/2- 21*dpi";
```

```
RadialMenu1->PointerPictureX = L"x + (width-pwidth)/2 + 1 * dpi";
RadialMenu1->AllowHotPointer = false;
RadialMenu1->SelForeColor[ExradialmenulibLib::RadialItemsEnum::exRadialFullItems] = RGB(0,0,0);
RadialMenu1->ParentSize = L"36*dpi";
RadialMenu1->ParentImageHeight[ExradialmenulibLib::RadialMenuStateEnum::exRadialMenuStateA = L"48*dpi";
RadialMenu1->ParentImageWidth[ExradialmenulibLib::RadialMenuStateEnum::exRadialMenuStateAl = L"48*dpi";
RadialMenu1->RadialLineSize[ExradialmenulibLib::RadialLineEnum::exRadialHotParent] = -1;
RadialMenu1->RadialLineAlpha[ExradialmenulibLib::RadialLineEnum::exRadialHotParent] = 32;
RadialMenu1->RadialLineColor[ExradialmenulibLib::RadialLineEnum::exRadialHotFullItem] = -1;
RadialMenu1->Expanded = true;
RadialMenu1->Items->ToString = L"Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8";
RadialMenu1->PointerIndex = 0;
RadialMenu1->SelectedIndex[ExradialmenulibLib::RadialItemsEnum::exRadialFullItems] = RadialMenu1->PointerIndex;
RadialMenu1->EndUpdate();
```

C#

```
// SelectItem event - Notifies once the user selects an item.
private void exradialmenu1_SelectItem(object sender,exontrol.EXRADIALMENULib.Item   Item)
{
   // PointerIndex = Item.Index

exradialmenu1.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRad
```

```
}
//this.exradialmenu1.SelectItem += new
exontrol.EXRADIALMENULib.exg2antt.SelectItemEventHandler(this.exradialmenu1_Selec


exradialmenu1.BeginUpdate();
exradialmenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
exradialmenu1.PointerPicture = "pointer.png";
exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
exradialmenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
exradialmenu1.AllowHotPointer = false;
exradialmenu1.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadia

exradialmenu1.ParentSize = "36*dpi";
exradialmenu1.set_ParentImageHeight(exontrol.EXRADIALMENULib.RadialMenuStateE

exradialmenu1.set_ParentImageWidth(exontrol.EXRADIALMENULib.RadialMenuStateEn

exradialmenu1.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadia

exradialmenu1.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRad

exradialmenu1.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exR

exradialmenu1.Expanded = true;
exradialmenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8";
exradialmenu1.PointerIndex = 0;
exradialmenu1.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRad

exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<SCRIPT FOR="RadialMenu1" EVENT="SelectItem(Item)" LANGUAGE="JScript">
   // PointerIndex = Item.Index
   RadialMenu1.SelectedIndex(3) = RadialMenu1.PointerIndex;
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
   RadialMenu1.PointerPicture = "pointer.png";
   RadialMenu1.SelBackAlpha(1) = 32;
   RadialMenu1.SelBackAlpha(2) = 128;
   RadialMenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
   RadialMenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
   RadialMenu1.AllowHotPointer = false;
   RadialMenu1.SelForeColor(3) = 0;
   RadialMenu1.ParentSize = "36*dpi";
   RadialMenu1.ParentImageHeight(-1) = "48*dpi";
   RadialMenu1.ParentImageWidth(-1) = "48*dpi";
   RadialMenu1.RadialLineSize(8) = -1;
   RadialMenu1.RadialLineAlpha(8) = 32;
   RadialMenu1.RadialLineColor(11) = -1;
   RadialMenu1.Expanded = true;
   RadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
   RadialMenu1.PointerIndex = 0;
   RadialMenu1.SelectedIndex(3) = RadialMenu1.PointerIndex;
```

```
    RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<SCRIPT LANGUAGE="VBScript">
Function RadialMenu1_SelectItem(Item)
    ' PointerIndex = Item.Index
    With RadialMenu1
        .SelectedIndex(3) = .PointerIndex
    End With
End Function
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
    With RadialMenu1
        .BeginUpdate
        .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
        .PointerPicture = "pointer.png"
        .SelBackAlpha(1) = 32
        .SelBackAlpha(2) = 128
        .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
        .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
        .AllowHotPointer = False
        .SelForeColor(3) = RGB(0,0,0)
        .ParentSize = "36*dpi"
        .ParentImageHeight(-1) = "48*dpi"
        .ParentImageWidth(-1) = "48*dpi"
        .RadialLineSize(8) = -1
```

```
      .RadialLineAlpha(8) = 32
      .RadialLineColor(11) = -1
      .Expanded = True
      .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
      .PointerIndex = 0
      .SelectedIndex(3) = .PointerIndex
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
// SelectItem event - Notifies once the user selects an item.
private void axRadialMenu1_SelectItem(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent e)
{
   // PointerIndex = Item.Index

axRadialMenu1.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullIte

}
//this.axRadialMenu1.SelectItem += new
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEventHandler(this.axRadialMenu1_



axRadialMenu1.BeginUpdate();
axRadialMenu1.PicturesPath = "C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images";
axRadialMenu1.PointerPicture = "pointer.png";
axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,3

axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubIte

axRadialMenu1.PointerPictureY = "y + (height-pheight)/2- 21*dpi";
```

```
axRadialMenu1.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi";
axRadialMenu1.AllowHotPointer = false;
axRadialMenu1.set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullIter
(uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0)));
axRadialMenu1.ParentSize = "36*dpi";
axRadialMenu1.set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exR

axRadialMenu1.set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRa

axRadialMenu1.set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotPar

axRadialMenu1.set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotP

axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFu

axRadialMenu1.Expanded = true;
axRadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
axRadialMenu1.PointerIndex = 0;
axRadialMenu1.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullIte

axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
// SelectItem event - Notifies once the user selects an item.
void onEvent_SelectItem(COM   _Item)
{
   // PointerIndex = Item.Index
   ;

exradialmenu1.SelectedIndex(3/*exRadialFullItems*/,exradialmenu1.PointerIndex());
}

public void init()
{
```

```
    ;

    super();

    exradialmenu1.BeginUpdate();
    exradialmenu1.PicturesPath("C:\\Program
Files\\Exontrol\\ExRadialMenu\\Sample\\Images");
    exradialmenu1.PointerPicture("pointer.png");
    exradialmenu1.SelBackAlpha(1/*exRadialItems*/,32);
    exradialmenu1.SelBackAlpha(2/*exRadialSubItems*/,128);
    exradialmenu1.PointerPictureY("y + (height-pheight)/2- 21*dpi");
    exradialmenu1.PointerPictureX("x + (width-pwidth)/2 + 1 * dpi");
    exradialmenu1.AllowHotPointer(false);
    exradialmenu1.SelForeColor(3/*exRadialFullItems*/,WinApi::RGB2int(0,0,0));
    exradialmenu1.ParentSize("36*dpi");
    exradialmenu1.ParentImageHeight(-1/*exRadialMenuStateAll*/,"48*dpi");
    exradialmenu1.ParentImageWidth(-1/*exRadialMenuStateAll*/,"48*dpi");
    exradialmenu1.RadialLineSize(8/*exRadialHotParent*/,-1);
    exradialmenu1.RadialLineAlpha(8/*exRadialHotParent*/,32);
    exradialmenu1.RadialLineColor(11/*exRadialHotFullItem*/,-1);
    exradialmenu1.Expanded(true);
    exradialmenu1.Items().ToString("Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8");
    exradialmenu1.PointerIndex(0);

exradialmenu1.SelectedIndex(3/*exRadialFullItems*/,exradialmenu1.PointerIndex());
    exradialmenu1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
// SelectItem event - Notifies once the user selects an item.
procedure TWinForm1.AxRadialMenu1_SelectItem(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent);
begin
    // PointerIndex = Item.Index
    with AxRadialMenu1 do
```

```
  begin

set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,PointerInde

  end
end;

with AxRadialMenu1 do
begin
  BeginUpdate();
  PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
  PointerPicture := 'pointer.png';
  set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32);
  set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128);
  PointerPictureY := 'y + (height-pheight)/2- 21*dpi';
  PointerPictureX := 'x + (width-pwidth)/2 + 1 * dpi';
  AllowHotPointer := False;
  set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,$0);
  ParentSize := '36*dpi';

set_ParentImageHeight(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuState


set_ParentImageWidth(EXRADIALMENULib.RadialMenuStateEnum.exRadialMenuStateA

  set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1);
  set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32);

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,$ffffffff);

  Expanded := True;
  Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
  PointerIndex := 0;

set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,PointerInde

  EndUpdate();
```

```
end
```

Delphi (standard)

```
// SelectItem event - Notifies once the user selects an item.
procedure TForm1.RadialMenu1SelectItem(ASender: TObject; Item : IItem);
begin
  // PointerIndex = Item.Index
  with RadialMenu1 do
  begin
    SelectedIndex[EXRADIALMENULib_TLB.exRadialFullItems] := PointerIndex;
  end
end;

with RadialMenu1 do
begin
  BeginUpdate();
  PicturesPath := 'C:\Program Files\Exontrol\ExRadialMenu\Sample\Images';
  PointerPicture := 'pointer.png';
  SelBackAlpha[EXRADIALMENULib_TLB.exRadialItems] := 32;
  SelBackAlpha[EXRADIALMENULib_TLB.exRadialSubItems] := 128;
  PointerPictureY := 'y + (height-pheight)/2- 21*dpi';
  PointerPictureX := 'x + (width-pwidth)/2 + 1 * dpi';
  AllowHotPointer := False;
  SelForeColor[EXRADIALMENULib_TLB.exRadialFullItems] := $0;
  ParentSize := '36*dpi';
  ParentImageHeight[EXRADIALMENULib_TLB.exRadialMenuStateAll] := '48*dpi';
  ParentImageWidth[EXRADIALMENULib_TLB.exRadialMenuStateAll] := '48*dpi';
  RadialLineSize[EXRADIALMENULib_TLB.exRadialHotParent] := -1;
  RadialLineAlpha[EXRADIALMENULib_TLB.exRadialHotParent] := 32;
  RadialLineColor[EXRADIALMENULib_TLB.exRadialHotFullItem] := $ffffffff;
  Expanded := True;
  Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
  PointerIndex := 0;
  SelectedIndex[EXRADIALMENULib_TLB.exRadialFullItems] := PointerIndex;
  EndUpdate();
end
```

VFP

```
*** SelectItem event - Notifies once the user selects an item. ***
LPARAMETERS Item
   *** PointerIndex = Item.Index
   with thisform.RadialMenu1
      .Object.SelectedIndex(3) = .PointerIndex
   endwith

with thisform.RadialMenu1
   .BeginUpdate
   .PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   .PointerPicture = "pointer.png"
   .Object.SelBackAlpha(1) = 32
   .Object.SelBackAlpha(2) = 128
   .PointerPictureY = "y + (height-pheight)/2- 21*dpi"
   .PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
   .AllowHotPointer = .F.
   .Object.SelForeColor(3) = RGB(0,0,0)
   .ParentSize = "36*dpi"
   .Object.ParentImageHeight(-1) = "48*dpi"
   .Object.ParentImageWidth(-1) = "48*dpi"
   .Object.RadialLineSize(8) = -1
   .Object.RadialLineAlpha(8) = 32
   .Object.RadialLineColor(11) = -1
   .Expanded = .T.
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
   .PointerIndex = 0
   .Object.SelectedIndex(3) = .PointerIndex
   .EndUpdate
endwith
```

dBASE Plus

```
/*
with (this.EXRADIALMENUACTIVEXCONTROL1.nativeObject)
   SelectItem = class::nativeObject_SelectItem
endwith
```

```
*/
// Notifies once the user selects an item.
function nativeObject_SelectItem(Item)
   /* PointerIndex = Item.Index */
   oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
   oRadialMenu.Template = [SelectedIndex(3) = PointerIndex] //
oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
return

local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.Template = [SelBackAlpha(1) = 32] // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = [SelBackAlpha(2) = 128] // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = false
oRadialMenu.Template = [SelForeColor(3) = 0] // oRadialMenu.SelForeColor(3) = 0x0
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.Template = [ParentImageHeight(-1) = "48*dpi"] //
oRadialMenu.ParentImageHeight(-1) = "48*dpi"
oRadialMenu.Template = [ParentImageWidth(-1) = "48*dpi"] //
oRadialMenu.ParentImageWidth(-1) = "48*dpi"
oRadialMenu.Template = [RadialLineSize(8) = -1] // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = [RadialLineAlpha(8) = 32] //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = [RadialLineColor(11) = -1] //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
```

```
8"
oRadialMenu.PointerIndex = 0
oRadialMenu.Template = [SelectedIndex(3) = PointerIndex] //
oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
' Notifies once the user selects an item.
function SelectItem as v (Item  as  OLE::Exontrol.RadialMenu.1::IItem)
   ' PointerIndex = Item.Index
   oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
   oRadialMenu.Template = "SelectedIndex(3) = PointerIndex" //
oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
end function

Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.Template = "SelBackAlpha(1) = 32" // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = "SelBackAlpha(2) = 128" // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = .f.
oRadialMenu.Template = "SelForeColor(3) = 0" // oRadialMenu.SelForeColor(3) = 0
oRadialMenu.ParentSize = "36*dpi"
oRadialMenu.Template = "ParentImageHeight(-1) = `48*dpi`" //
oRadialMenu.ParentImageHeight(-1) = "48*dpi"
oRadialMenu.Template = "ParentImageWidth(-1) = `48*dpi`" //
oRadialMenu.ParentImageWidth(-1) = "48*dpi"
```

```
oRadialMenu.Template = "RadialLineSize(8) = -1" // oRadialMenu.RadialLineSize(8) = -1
oRadialMenu.Template = "RadialLineAlpha(8) = 32" // oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = "RadialLineColor(11) = -1" // oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = .t.
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
oRadialMenu.PointerIndex = 0
oRadialMenu.Template = "SelectedIndex(3) = PointerIndex" // oRadialMenu.SelectedIndex(3) = oRadialMenu.PointerIndex
oRadialMenu.EndUpdate()
```

Visual Objects

```
METHOD OCX_Exontrol1SelectItem(Item) CLASS MainDialog
   // SelectItem event - Notifies once the user selects an item.
   // PointerIndex = Item.Index
   oDCOCX_Exontrol1:[SelectedIndex,exRadialFullItems] :=
oDCOCX_Exontrol1:PointerIndex
RETURN NIL



oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:PicturesPath := "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
oDCOCX_Exontrol1:PointerPicture := "pointer.png"
oDCOCX_Exontrol1:[SelBackAlpha,exRadialItems] := 32
oDCOCX_Exontrol1:[SelBackAlpha,exRadialSubItems] := 128
oDCOCX_Exontrol1:PointerPictureY := "y + (height-pheight)/2- 21*dpi"
oDCOCX_Exontrol1:PointerPictureX := "x + (width-pwidth)/2 + 1 * dpi"
oDCOCX_Exontrol1:AllowHotPointer := false
oDCOCX_Exontrol1:[SelForeColor,exRadialFullItems] := RGB(0,0,0)
oDCOCX_Exontrol1:ParentSize := "36*dpi"
oDCOCX_Exontrol1:[ParentImageHeight,exRadialMenuStateAll] := "48*dpi"
```

```
oDCOCX_Exontrol1:[ParentImageWidth,exRadialMenuStateAll] := "48*dpi"
oDCOCX_Exontrol1:[RadialLineSize,exRadialHotParent] := -1
oDCOCX_Exontrol1:[RadialLineAlpha,exRadialHotParent] := 32
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotFullItem] := -1
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:Items:ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
oDCOCX_Exontrol1:PointerIndex := 0
oDCOCX_Exontrol1:[SelectedIndex,exRadialFullItems] :=
oDCOCX_Exontrol1:PointerIndex
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
/*begin event SelectItem(oleobject  Item) - Notifies once the user selects an item.*/
/*
   PointerIndex = Item.Index
   oRadialMenu = ole_1.Object
   oRadialMenu.SelectedIndex(3,oRadialMenu.PointerIndex)
*/
/*end event SelectItem*/

OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.PicturesPath = "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
oRadialMenu.PointerPicture = "pointer.png"
oRadialMenu.SelBackAlpha(1,32)
oRadialMenu.SelBackAlpha(2,128)
oRadialMenu.PointerPictureY = "y + (height-pheight)/2- 21*dpi"
oRadialMenu.PointerPictureX = "x + (width-pwidth)/2 + 1 * dpi"
oRadialMenu.AllowHotPointer = false
oRadialMenu.SelForeColor(3,RGB(0,0,0))
oRadialMenu.ParentSize = "36*dpi"
```

```
oRadialMenu.ParentImageHeight(-1,"48*dpi")
oRadialMenu.ParentImageWidth(-1,"48*dpi")
oRadialMenu.RadialLineSize(8,-1)
oRadialMenu.RadialLineAlpha(8,32)
oRadialMenu.RadialLineColor(11,-1)
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.PointerIndex = 0
oRadialMenu.SelectedIndex(3,oRadialMenu.PointerIndex)
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
// Notifies once the user selects an item.
Procedure OnComSelectItem Variant   llItem
   Forward Send OnComSelectItem llItem
   // PointerIndex = Item.Index
   Set ComSelectedIndex OLEexRadialFullItems to (ComPointerIndex(Self))
End_Procedure

Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComPicturesPath to "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
   Set ComPointerPicture to "pointer.png"
   Set ComSelBackAlpha OLEexRadialItems to 32
   Set ComSelBackAlpha OLEexRadialSubItems to 128
   Set ComPointerPictureY to "y + (height-pheight)/2- 21*dpi"
   Set ComPointerPictureX to "x + (width-pwidth)/2 + 1 * dpi"
   Set ComAllowHotPointer to False
   Set ComSelForeColor OLEexRadialFullItems to (RGB(0,0,0))
   Set ComParentSize to "36*dpi"
   Set ComParentImageHeight OLEexRadialMenuStateAll to "48*dpi"
   Set ComParentImageWidth OLEexRadialMenuStateAll to "48*dpi"
   Set ComRadialLineSize OLEexRadialHotParent to -1
```

```
    Set ComRadialLineAlpha OLEexRadialHotParent to 32
    Set ComRadialLineColor OLEexRadialHotFullItem to -1
    Set ComExpanded to True
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Set ComToString of hoItems to "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
    Send Destroy to hoItems
    Set ComPointerIndex to 0
    Set ComSelectedIndex OLEexRadialFullItems to (ComPointerIndex(Self))
    Send ComEndUpdate
End_Procedure
```

XBase++

```
PROCEDURE OnSelectItem(oRadialMenu,Item)
   /*PointerIndex = Item.Index*/

oRadialMenu:SetProperty("SelectedIndex",3/*exRadialFullItems*/,oRadialMenu:Pointe

RETURN

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

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

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

```
   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:SelectItem := {|Item| OnSelectItem(oRadialMenu,Item)} /*Notifies
once the user selects an item.*/

      oRadialMenu:BeginUpdate()
      oRadialMenu:PicturesPath := "C:\Program
Files\Exontrol\ExRadialMenu\Sample\Images"
      oRadialMenu:PointerPicture := "pointer.png"
      oRadialMenu:SetProperty("SelBackAlpha",1/*exRadialItems*/,32)
      oRadialMenu:SetProperty("SelBackAlpha",2/*exRadialSubItems*/,128)
      oRadialMenu:PointerPictureY := "y + (height-pheight)/2- 21*dpi"
      oRadialMenu:PointerPictureX := "x + (width-pwidth)/2 + 1 * dpi"
      oRadialMenu:AllowHotPointer := .F.

oRadialMenu:SetProperty("SelForeColor",3/*exRadialFullItems*/,AutomationTranslateC
 GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oRadialMenu:ParentSize := "36*dpi"

oRadialMenu:SetProperty("ParentImageHeight",-1/*exRadialMenuStateAll*/,"48*dpi")

oRadialMenu:SetProperty("ParentImageWidth",-1/*exRadialMenuStateAll*/,"48*dpi")
      oRadialMenu:SetProperty("RadialLineSize",8/*exRadialHotParent*/,-1)
      oRadialMenu:SetProperty("RadialLineAlpha",8/*exRadialHotParent*/,32)
      oRadialMenu:SetProperty("RadialLineColor",11/*exRadialHotFullItem*/,-1)
      oRadialMenu:Expanded := .T.
      oRadialMenu:Items():ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
      oRadialMenu:PointerIndex := 0

oRadialMenu:SetProperty("SelectedIndex",3/*exRadialFullItems*/,oRadialMenu:Pointe

      oRadialMenu:EndUpdate()
```

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

property RadialMenu.PointerPictureHeight as String

Specifies the height of the the pointer, relative to the center of the radial menu.

Type	Description
String	A String expression that defines the height of the pointer, relative to the center of the radial menu.

By default, the PointerPictureWidth property is "pheight", which indicates that the pointer picture is displayed as it is loaded, not stretching. The PointerPictureHeight property specifies the height of the pointer picture. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The SelectedIndex property specifies the index of the item/slice to be selected.

The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

The PointerIndex property specifies the index within the radial menu to target the pointer. The PointerAngle property specifies the angle of the pointer to target another item or index.

The PointerPictureHeight property supports the following keywords:

- **pwidth**, indicates the width in pixels of the pointer picture
- **pheight**, indicates the height in pixels of the pointer picture
- **width**, indicates the width in pixels of the control.
- **height**, indicates the height in pixels of the control.
- **x**, specifies the x-coordinate of the center.
- **y**, specifies the y-coordinate of the center.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if

it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.

- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.

- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the

result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.PointerPictureWidth as String

Specifies the width of the the pointer, relative to the center of the radial menu.

Type	Description
String	A String expression that defines width of the pointer, relative to the center of the radial menu.

By default, the PointerPictureWidth property is "pwidth", which indicates that the pointer picture is displayed as it is loaded, not stretching. The PointerPictureWidth property specifies the width of the pointer picture. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The SelectedIndex property specifies the index of the item/slice to be selected.

The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

The PointerIndex property specifies the index within the radial menu to target the pointer. The PointerAngle property specifies the angle of the pointer to target another item or index.

The PointerPictureWidth property supports the following keywords:

- **pwidth**, indicates the width in pixels of the pointer picture
- **pheight**, indicates the height in pixels of the pointer picture
- **width**, indicates the width in pixels of the control.
- **height**, indicates the height in pixels of the control.
- **x**, specifies the x-coordinate of the center.
- **y**, specifies the y-coordinate of the center.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if

it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.

- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.

- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the

result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.PointerPictureX as String

Specifies the x-coordinate of the the pointer, relative to the center of the radial menu.

Type	Description
String	A String expression that defines the x-coordinate of the the pointer, relative to the center of the radial menu.

By default, the PointerPictureX property is "x + (width-pwidth)/2", which indicates that the pointer picture is displayed in the center of the control. The PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The SelectedIndex property specifies the index of the item/slice to be selected.

The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

The PointerIndex property specifies the index within the radial menu to target the pointer. The PointerAngle property specifies the angle of the pointer to target another item or index.

The PointerPictureX property supports the following keywords:

- **pwidth**, indicates the width in pixels of the pointer picture
- **pheight**, indicates the height in pixels of the pointer picture
- **width**, indicates the width in pixels of the control.
- **height**, indicates the height in pixels of the control.
- **x**, specifies the x-coordinate of the center.
- **y**, specifies the y-coordinate of the center.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if

it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.

- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.

- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the

result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.PointerPictureY as String

Specifies the y-coordinate of the the pointer, relative to the center of the radial menu.

Type	Description
String	A String expression that defines the y-coordinate of the the pointer, relative to the center of the radial menu.

By default, the PointerPictureY property is "y + (height-pheight)/2", which indicates that the pointer picture is displayed in the center of the control. The PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The AllowHotPointer property indicates whether the pointer is oriented to the item, while hovering the radial menu. No pointer is shown, while the PointerPicture property is empty, even if the AllowHotPointer property is True. The SelectedIndex property specifies the index of the item/slice to be selected.

The following properties specifies where the pointer picture should be placed, relative to the center of the radial menu:

- PointerPictureX property specifies the x-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureY property specifies the y-coordinate of the the pointer, relative to the center of the radial menu.
- PointerPictureWidth property specifies the width of the the pointer, relative to the center of the radial menu.
- PointerPictureHeight property specifies the height of the the pointer, relative to the center of the radial menu.

The PointerIndex property specifies the index within the radial menu to target the pointer. The PointerAngle property specifies the angle of the pointer to target another item or index.

The PointerPictureY property supports the following keywords:

- **pwidth**, indicates the width in pixels of the pointer picture
- **pheight**, indicates the height in pixels of the pointer picture
- **width**, indicates the width in pixels of the control.
- **height**, indicates the height in pixels of the control.
- **x**, specifies the x-coordinate of the center.
- **y**, specifies the y-coordinate of the center.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if

it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.

- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.

- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the

result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.RadialLineAlpha(Line as RadialLineEnum) as Byte

Specifies the value of alpha / opacity channel to show the giving line within the radial menu.

Type	Description
Line as RadialLineEnum	A RadialLineEnum expression that specifies the line/border to be changed.
Byte	A BYTE expression that specifies the value of alpha / opacity channel to show the radial line. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

The RadialLineAlpha property specifies the value of alpha / opacity channel to show the giving line within the radial menu. The RadialLineColor property specifies the color to show the given radial line within the control. The RadialLineSize property specifies the size to show the giving line within the radial menu. The RadialLineStyle property specifies the style to show the given radial line within the control.

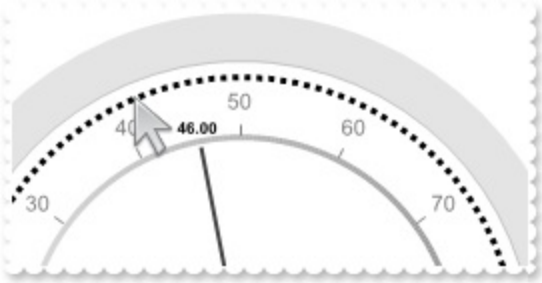
property RadialMenu.RadialLineColor(Line as RadialLineEnum) as Color

Specifies the color to show the given radial line within the control.

Type	Description
Line as RadialLineEnum	A RadialLineEnum expression that specifies the line/border to be changed.
Color	A Color expression that specifies the color to be applied to the radial-line. If -1, the line is hidden.

The RadialLineColor property specifies the color to show the given radial line within the control. The RadialLineAlpha property specifies the value of alpha / opacity channel to show the giving line within the radial menu. The RadialLineSize property specifies the size to show the giving line within the radial menu. The RadialLineStyle property specifies the style to show the given radial line within the control. The ArrowImage property specifies the graphics (image, icon, picture) to be shown on the sub-items zone, for items that contains child items or sub items.

The following screen shot show the exRadialCustomBorder, with a different style and size:



property RadialMenu.RadialLineSize(Line as RadialLineEnum) as Long

Specifies the size to show the giving line within the radial menu.

Type	Description
Line as RadialLineEnum	A RadialLineEnum expression that specifies the line/border to be changed.
Long	A Long expression that specifies the size to display the radial line. The hot radial-lines supports -1, which indicates that the indicate region is filled.

The RadialLineSize property specifies the size to show the giving line within the radial menu. The RadialLineColor property specifies the color to show the given radial line within the control. The RadialLineAlpha property specifies the value of alpha / opacity channel to show the giving line within the radial menu. The RadialLineStyle property specifies the style to show the given radial line within the control.

property RadialMenu.RadialLineStyle(Line as RadialLineEnum) as RadialLineStyleEnum

Specifies the style to show the given radial line within the control.

Type	Description
Line as RadialLineEnum	A RadialLineEnum expression that specifies the line/border to be changed.
RadialLineStyleEnum	A RadialLineStyleEnum expression that specifies the style of the radial line.

The RadialLineStyle property specifies the style to show the given radial line within the control. The RadialLineColor property specifies the color to show the given radial line within the control. The RadialLineAlpha property specifies the value of alpha / opacity channel to show the giving line within the radial menu. The RadialLineSize property specifies the size to show the giving line within the radial menu.

method RadialMenu.Refresh ()

Refreshes the control.

Type	Description

method RadialMenu.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 = ExRadialMenu1.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 = ExRadialMenu1.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:

```
 ExRadialMenu1.ReplaceIcon 0,  i, where i specifies the index of icon removed.
```

The following VB clears the control's icons collection:

```
 ExRadialMenu1.ReplaceIcon 0,  -1
```

property RadialMenu.Root as Item

Retrieves the root item.

Type	Description
Item	An Item object that specifies the root of the control.

The Root property of the control accesses the root item. The root item has no parent item. The Parent item property specifies the parent item. The Add method adds new items (child) to the control. The Items property accesses the child-items collection of the current item. The SelectParent event occurs once the user clicks the parent of the item. The BrowseItem event notifies when a new item has been selected / browsed.

property RadialMenu.SelBackAlpha(Type as RadialItemsEnum) as Byte

Specifies the value of alpha / opacity channel to show the selection of the radial menu.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that determines what portion of the item is selected.
Byte	A BYTE expression that specifies the value of alpha / opacity channel to show the selection. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

By default, the SelBackAlpha property is 255. The SelBackColor / SelBackAlpha property specifies the selection background color. The SelForeColor property specifies the selection foreground color. The SelectedIndex property gets or sets a value that indicates index to be selected. The SelectItem event notifies once the user selects an item. The SelectItem event is fired when user clicks an item with no child items. The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The SelectParent event occurs once the user clicks the parent of the item. The PointerPicture property indicates the picture to be shown on the pointer zone's background.

property RadialMenu.SelBackColor(Type as RadialItemsEnum) as Color

Specifies the selection background color.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that determines what portion of the item is selected.
Color	A Color expression that defines the color to be applied. If -1 no color is applied.

By default, the SelBackColor property is -1, so no color is applied for selected index. The SelBackColor / SelBackAlpha property specifies the selection background color. The SelForeColor property specifies the selection foreground color. The SelectedIndex property gets or sets a value that indicates index to be selected. The SelectItem event notifies once the user selects an item. The SelectItem event is fired when user clicks an item with no child items. The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The SelectParent event occurs once the user clicks the parent of the item. The PointerPicture property indicates the picture to be shown on the pointer zone's background.

property RadialMenu.SelectedIndex(Type as RadialItemsEnum) as Long

Gets or sets a value that indicates index to be selected.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that determines what portion of the item is selected.
Long	A Long expression that specifies the index of the item being selected.

The SelectedIndex property gets or sets a value that indicates index to be selected. The SelectItem event notifies once the user selects an item. The SelectItem event is fired when user clicks an item with no child items. The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The SelectParent event occurs once the user clicks the parent of the item. The PointerPicture property indicates the picture to be shown on the pointer zone's background. The SelBackColor / SelBackAlpha property specifies the selection background color. The SelForeColor property specifies the selection foreground color.

The following sample shows how you can select an item, once the user clicks it:

VBA (MS Access, Excell...)

```
' SelectItem event - Notifies once the user selects an item.
Private Sub RadialMenu1_SelectItem(ByVal Item As Object)
    ' SelectedIndex(3) = Item.Index
End Sub

With RadialMenu1
    .BeginUpdate
    .SelBackAlpha(1) = 32
    .SelBackAlpha(2) = 128
    .SelForeColor(3) = RGB(0,0,0)
    .RadialLineSize(8) = -1
    .RadialLineAlpha(8) = 32
    .RadialLineColor(11) = -1
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .SelectedIndex(3) = 0
```

```
    .EndUpdate
End With
```

VB6

```
' SelectItem event - Notifies once the user selects an item.
Private Sub RadialMenu1_SelectItem(ByVal Item As EXRADIALMENULibCtl.IItem)
    ' SelectedIndex(3) = Item.Index
End Sub

With RadialMenu1
    .BeginUpdate
    .SelBackAlpha(exRadialItems) = 32
    .SelBackAlpha(exRadialSubItems) = 128
    .SelForeColor(exRadialFullItems) = RGB(0,0,0)
    .RadialLineSize(exRadialHotParent) = -1
    .RadialLineAlpha(exRadialHotParent) = 32
    .RadialLineColor(exRadialHotFullItem) = -1
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .SelectedIndex(exRadialFullItems) = 0
    .EndUpdate
End With
```

VB.NET

```
' SelectItem event - Notifies once the user selects an item.
Private Sub Exradialmenu1_SelectItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXRADIALMENULib.Item) Handles Exradialmenu1.SelectItem
    ' SelectedIndex(3) = Item.Index
End Sub

With Exradialmenu1
    .BeginUpdate()
    .set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)

.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128
```

```
.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,Colo

.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)

.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,3

.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotFullIte

    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"

.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)

    .EndUpdate()
End With
```

VB.NET for /COM

```
' SelectItem event - Notifies once the user selects an item.
Private Sub AxRadialMenu1_SelectItem(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent) Handles
AxRadialMenu1.SelectItem
    ' SelectedIndex(3) = Item.Index
End Sub

With AxRadialMenu1
    .BeginUpdate()
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128)
    .set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)
    .set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)
    .set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32)
    .set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,-1)
```

```
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)
    .EndUpdate()
End With
```

C++

```
// SelectItem event - Notifies once the user selects an item.
void OnSelectItemRadialMenu1(LPDISPATCH  Item)
{
   // SelectedIndex(3) = Item.Index
}

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialItems,32);
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialSubItems,128);
spRadialMenu1->PutSelForeColor(EXRADIALMENULib::exRadialFullItems,RGB(0,0,0));
spRadialMenu1->PutRadialLineSize(EXRADIALMENULib::exRadialHotParent,-1);
spRadialMenu1->PutRadialLineAlpha(EXRADIALMENULib::exRadialHotParent,32);
spRadialMenu1->PutRadialLineColor(EXRADIALMENULib::exRadialHotFullItem,-1);
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->GetItems()->PutToString(L"Item 1,Item 2,Item 3,Item 4,Item 5,Item
6,Item 7,Item 8");
spRadialMenu1->PutSelectedIndex(EXRADIALMENULib::exRadialFullItems,0);
spRadialMenu1->EndUpdate();
```

C++ Builder

```
// SelectItem event - Notifies once the user selects an item.
void __fastcall TForm1::RadialMenu1SelectItem(TObject
*Sender,Exradialmenulib_tlb::IItem  *Item)
{
  // SelectedIndex(3) = Item.Index
}

RadialMenu1->BeginUpdate();
RadialMenu1->SelBackAlpha[Exradialmenulib_tlb::RadialItemsEnum::exRadialItems] =
32;
RadialMenu1-
>SelBackAlpha[Exradialmenulib_tlb::RadialItemsEnum::exRadialSubItems] = 128;
RadialMenu1-
>SelForeColor[Exradialmenulib_tlb::RadialItemsEnum::exRadialFullItems] =
RGB(0,0,0);
RadialMenu1-
>RadialLineSize[Exradialmenulib_tlb::RadialLineEnum::exRadialHotParent] = -1;
RadialMenu1-
>RadialLineAlpha[Exradialmenulib_tlb::RadialLineEnum::exRadialHotParent] = 32;
RadialMenu1-
>RadialLineColor[Exradialmenulib_tlb::RadialLineEnum::exRadialHotFullItem] = -1;
RadialMenu1->Expanded = true;
RadialMenu1->Items->ToString = L"Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
RadialMenu1-
>SelectedIndex[Exradialmenulib_tlb::RadialItemsEnum::exRadialFullItems] = 0;
RadialMenu1->EndUpdate();
```

C#

```
// SelectItem event - Notifies once the user selects an item.
private void exradialmenu1_SelectItem(object
sender,exontrol.EXRADIALMENULib.Item  Item)
{
  // SelectedIndex(3) = Item.Index
```

```
}
//this.exradialmenu1.SelectItem += new
exontrol.EXRADIALMENULib.exg2antt.SelectItemEventHandler(this.exradialmenu1_Selec


exradialmenu1.BeginUpdate();
exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadia

exradialmenu1.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadia

exradialmenu1.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRad

exradialmenu1.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exR

exradialmenu1.Expanded = true;
exradialmenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8";
exradialmenu1.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRa

exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<SCRIPT FOR="RadialMenu1" EVENT="SelectItem(Item)" LANGUAGE="JScript">
  // SelectedIndex(3) = Item.Index
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
```

```
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.SelBackAlpha(1) = 32;
   RadialMenu1.SelBackAlpha(2) = 128;
   RadialMenu1.SelForeColor(3) = 0;
   RadialMenu1.RadialLineSize(8) = -1;
   RadialMenu1.RadialLineAlpha(8) = 32;
   RadialMenu1.RadialLineColor(11) = -1;
   RadialMenu1.Expanded = true;
   RadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
   RadialMenu1.SelectedIndex(3) = 0;
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<SCRIPT LANGUAGE="VBScript">
Function RadialMenu1_SelectItem(Item)
   ' SelectedIndex(3) = Item.Index
End Function
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .SelBackAlpha(1) = 32
      .SelBackAlpha(2) = 128
```

```
      .SelForeColor(3) = RGB(0,0,0)
      .RadialLineSize(8) = -1
      .RadialLineAlpha(8) = 32
      .RadialLineColor(11) = -1
      .Expanded = True
      .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
      .SelectedIndex(3) = 0
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
// SelectItem event - Notifies once the user selects an item.
private void axRadialMenu1_SelectItem(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent e)
{
   // SelectedIndex(3) = Item.Index
}
//this.axRadialMenu1.SelectItem += new
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEventHandler(this.axRadialMenu1_


axRadialMenu1.BeginUpdate();
axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,3

axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubIte

axRadialMenu1.set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullIter
(uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0)));
axRadialMenu1.set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotPar

axRadialMenu1.set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotP
```

```
axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFu

axRadialMenu1.Expanded = true;
axRadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
axRadialMenu1.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullI

axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
// SelectItem event - Notifies once the user selects an item.
void onEvent_SelectItem(COM   _Item)
{
   // SelectedIndex(3) = Item.Index
   ;
}

public void init()
{
   ;

   super();

   exradialmenu1.BeginUpdate();
   exradialmenu1.SelBackAlpha(1/*exRadialItems*/,32);
   exradialmenu1.SelBackAlpha(2/*exRadialSubItems*/,128);
   exradialmenu1.SelForeColor(3/*exRadialFullItems*/,WinApi::RGB2int(0,0,0));
   exradialmenu1.RadialLineSize(8/*exRadialHotParent*/,-1);
   exradialmenu1.RadialLineAlpha(8/*exRadialHotParent*/,32);
   exradialmenu1.RadialLineColor(11/*exRadialHotFullItem*/,-1);
   exradialmenu1.Expanded(true);
   exradialmenu1.Items().ToString("Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8");
   exradialmenu1.SelectedIndex(3/*exRadialFullItems*/,0);
   exradialmenu1.EndUpdate();
```

```
}
```

Delphi 8 (.NET only)

```
// SelectItem event - Notifies once the user selects an item.
procedure TWinForm1.AxRadialMenu1_SelectItem(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent);
begin
   // SelectedIndex(3) = Item.Index
end;

with AxRadialMenu1 do
begin
   BeginUpdate();
   set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32);
   set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128);
   set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,$0);
   set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1);
   set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32);

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,$ffffffff);

   Expanded := True;
   Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
   set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0);
   EndUpdate();
end
```

Delphi (standard)

```
// SelectItem event - Notifies once the user selects an item.
procedure TForm1.RadialMenu1SelectItem(ASender: TObject; Item : IItem);
begin
   // SelectedIndex(3) = Item.Index
end;

with RadialMenu1 do
begin
```

```
  BeginUpdate();
  SelBackAlpha[EXRADIALMENULib_TLB.exRadialItems] := 32;
  SelBackAlpha[EXRADIALMENULib_TLB.exRadialSubItems] := 128;
  SelForeColor[EXRADIALMENULib_TLB.exRadialFullItems] := $0;
  RadialLineSize[EXRADIALMENULib_TLB.exRadialHotParent] := -1;
  RadialLineAlpha[EXRADIALMENULib_TLB.exRadialHotParent] := 32;
  RadialLineColor[EXRADIALMENULib_TLB.exRadialHotFullItem] := $ffffffff;
  Expanded := True;
  Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
  SelectedIndex[EXRADIALMENULib_TLB.exRadialFullItems] := 0;
  EndUpdate();
end
```

VFP

```
*** SelectItem event - Notifies once the user selects an item. ***
LPARAMETERS Item
   *** SelectedIndex(3) = Item.Index

with thisform.RadialMenu1
   .BeginUpdate
   .Object.SelBackAlpha(1) = 32
   .Object.SelBackAlpha(2) = 128
   .Object.SelForeColor(3) = RGB(0,0,0)
   .Object.RadialLineSize(8) = -1
   .Object.RadialLineAlpha(8) = 32
   .Object.RadialLineColor(11) = -1
   .Expanded = .T.
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
   .Object.SelectedIndex(3) = 0
   .EndUpdate
endwith
```

dBASE Plus

```
/*
with (this.EXRADIALMENUACTIVEXCONTROL1.nativeObject)
   SelectItem = class::nativeObject_SelectItem
```

```
endwith
*/
// Notifies once the user selects an item.
function nativeObject_SelectItem(Item)
   /* SelectedIndex(3) = Item.Index */
   oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
return

local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Template = [SelBackAlpha(1) = 32] // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = [SelBackAlpha(2) = 128] // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.Template = [SelForeColor(3) = 0] // oRadialMenu.SelForeColor(3) = 0x0
oRadialMenu.Template = [RadialLineSize(8) = -1] // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = [RadialLineAlpha(8) = 32] //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = [RadialLineColor(11) = -1] //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.Template = [SelectedIndex(3) = 0] // oRadialMenu.SelectedIndex(3) = 0
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
' Notifies once the user selects an item.
function SelectItem as v (Item  as  OLE::Exontrol.RadialMenu.1::IItem)
   ' SelectedIndex(3) = Item.Index
   oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
end function
```

```
Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Template = "SelBackAlpha(1) = 32" // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = "SelBackAlpha(2) = 128" // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.Template = "SelForeColor(3) = 0" // oRadialMenu.SelForeColor(3) = 0
oRadialMenu.Template = "RadialLineSize(8) = -1" // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = "RadialLineAlpha(8) = 32" //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = "RadialLineColor(11) = -1" //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = .t.
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.Template = "SelectedIndex(3) = 0" // oRadialMenu.SelectedIndex(3) = 0
oRadialMenu.EndUpdate()
```

Visual Objects

```
METHOD OCX_Exontrol1SelectItem(Item) CLASS MainDialog
   // SelectItem event - Notifies once the user selects an item.
   // SelectedIndex(3) = Item.Index

RETURN NIL



oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:[SelBackAlpha,exRadialItems] := 32
oDCOCX_Exontrol1:[SelBackAlpha,exRadialSubItems] := 128
oDCOCX_Exontrol1:[SelForeColor,exRadialFullItems] := RGB(0,0,0)
oDCOCX_Exontrol1:[RadialLineSize,exRadialHotParent] := -1
```

```
oDCOCX_Exontrol1:[RadialLineAlpha,exRadialHotParent] := 32
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotFullItem] := -1
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:Items:ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
oDCOCX_Exontrol1:[SelectedIndex,exRadialFullItems] := 0
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
/*begin event SelectItem(oleobject  Item) - Notifies once the user selects an item.*/
/*
    SelectedIndex(3) = Item.Index
    oRadialMenu = ole_1.Object
*/
/*end event SelectItem*/

OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.SelBackAlpha(1,32)
oRadialMenu.SelBackAlpha(2,128)
oRadialMenu.SelForeColor(3,RGB(0,0,0))
oRadialMenu.RadialLineSize(8,-1)
oRadialMenu.RadialLineAlpha(8,32)
oRadialMenu.RadialLineColor(11,-1)
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
oRadialMenu.SelectedIndex(3,0)
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
// Notifies once the user selects an item.
```

```
Procedure OnComSelectItem Variant   llItem
   Forward Send OnComSelectItem llItem
   // SelectedIndex(3) = Item.Index
End_Procedure

Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComSelBackAlpha OLEexRadialItems to 32
   Set ComSelBackAlpha OLEexRadialSubItems to 128
   Set ComSelForeColor OLEexRadialFullItems to (RGB(0,0,0))
   Set ComRadialLineSize OLEexRadialHotParent to -1
   Set ComRadialLineAlpha OLEexRadialHotParent to 32
   Set ComRadialLineColor OLEexRadialHotFullItem to -1
   Set ComExpanded to True
   Variant voItems
   Get ComItems to voItems
   Handle hoItems
   Get Create (RefClass(cComItems)) to hoItems
   Set pvComObject of hoItems to voItems
      Set ComToString of hoItems to "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
   Send Destroy to hoItems
   Set ComSelectedIndex OLEexRadialFullItems to 0
   Send ComEndUpdate
End_Procedure
```

XBase++

```
PROCEDURE OnSelectItem(oRadialMenu,Item)
   /*SelectedIndex(3) = Item.Index*/

RETURN

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

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

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:SelectItem := {|Item| OnSelectItem(oRadialMenu,Item)} /*Notifies
once the user selects an item.*/

      oRadialMenu:BeginUpdate()
      oRadialMenu:SetProperty("SelBackAlpha",1/*exRadialItems*/,32)
      oRadialMenu:SetProperty("SelBackAlpha",2/*exRadialSubItems*/,128)

oRadialMenu:SetProperty("SelForeColor",3/*exRadialFullItems*/,AutomationTranslateC
 GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oRadialMenu:SetProperty("RadialLineSize",8/*exRadialHotParent*/,-1)
      oRadialMenu:SetProperty("RadialLineAlpha",8/*exRadialHotParent*/,32)
      oRadialMenu:SetProperty("RadialLineColor",11/*exRadialHotFullItem*/,-1)
      oRadialMenu:Expanded := .T.
      oRadialMenu:Items():ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
      oRadialMenu:SetProperty("SelectedIndex",3/*exRadialFullItems*/,0)
      oRadialMenu:EndUpdate()

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

```
    ENDDO
RETURN
```

property RadialMenu.SelForeColor(Type as RadialItemsEnum) as Color

Specifies the selection foreground color.

Type	Description
Type as RadialItemsEnum	A RadialItemsEnum expression that determines what portion of the item is selected.
Color	A Color expression that defines the color to be applied.

The SelForeColor property specifies the selection foreground color. The SelBackColor / SelBackAlpha property specifies the selection background color. The SelectedIndex property gets or sets a value that indicates index to be selected. The SelectItem event notifies once the user selects an item. The SelectItem event is fired when user clicks an item with no child items. The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The SelectParent event occurs once the user clicks the parent of the item. The PointerPicture property indicates the picture to be shown on the pointer zone's background.

property RadialMenu.ShadowColor as Color

Specifies the control's shadow color.

Type	Description
Color	A Color expression that defines the color to show the control's shadow. If -1 no shadow is applied.

By default, the ShadowColor property is RGB(196,196,196). The ShadowColor property specifies the control's shadow color. You can hide the control's shadow, if using the ShadowColor property on -1. The SubItemsSize property specifies the size to display the sub-items zone. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu. The BackgroundPicture property indicates the picture to be shown on the radial menu's background.

property RadialMenu.ShowImageList as Boolean

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

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

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

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

Shows the specified tooltip at given position.

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

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

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

For instance:

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

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

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

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

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

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

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

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

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

property RadialMenu.State as RadialMenuStateEnum

Specifies the state of the radial menu.

Type	Description
RadialMenuStateEnum	A RadialMenuStateEnum expression that specifies the current state of the control.

By default, the State property is exRadialMenuCollapsed, which indicates the radial menu is shown as collapsed. The State property specifies the state of the radial menu. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Expanded property indicates whether the radial menu is expanded or collapsed. The ParentCaption property specifies the caption to be displayed on the parent portion of the control, based on the radial menu's state. The ParentImage property specifies the graphics (image, icon, picture) to be shown on the parent zone, based on the state of the radial menu. The ParentImageWidth / ParentImageHeight specifies the size to show the parent image, based on the radial menu's state.

property RadialMenu.SubItemsBackAlpha as Byte

Specifies the value of alpha / opacity channel to show the sub items zone of the radial menu.

Type	Description
Byte	A BYTE expression that specifies the value of alpha / opacity channel to show the sub-items portion of the radial menu. 255 indicates opaque, 128 semi-transparent, and 0 fully transparent.

By default, The SubItemsBackAlpha property is 255. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub- items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The RadialLineColor(exRadialSubItemsBorder) property specifies the color to show the border around the sub-items section of the control. The RadialLineColor(exRadialSubItemsGridLines) property specifies the color to show the grid lines, inside the sub-items section of the control.

property RadialMenu.SubItemsBackColor as Color

Specifies the color to show the sub items zone of the radial menu.

Type	Description
Color	A Color expression that specifies the color to show the sub-items portion of the radial menu. If -1, no solid color is applied on the sub-items portion of the control.

By default, The SubItemsBackColor property is RGB(228,228,228). The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub-items zone of the radial menu. The SubItemsSize property specifies the size to display the sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The RadialLineColor(exRadialSubItemsBorder) property specifies the color to show the border around the sub-items section of the control. The RadialLineColor(exRadialSubItemsGridLines) property specifies the color to show the grid lines, inside the sub-items section of the control.

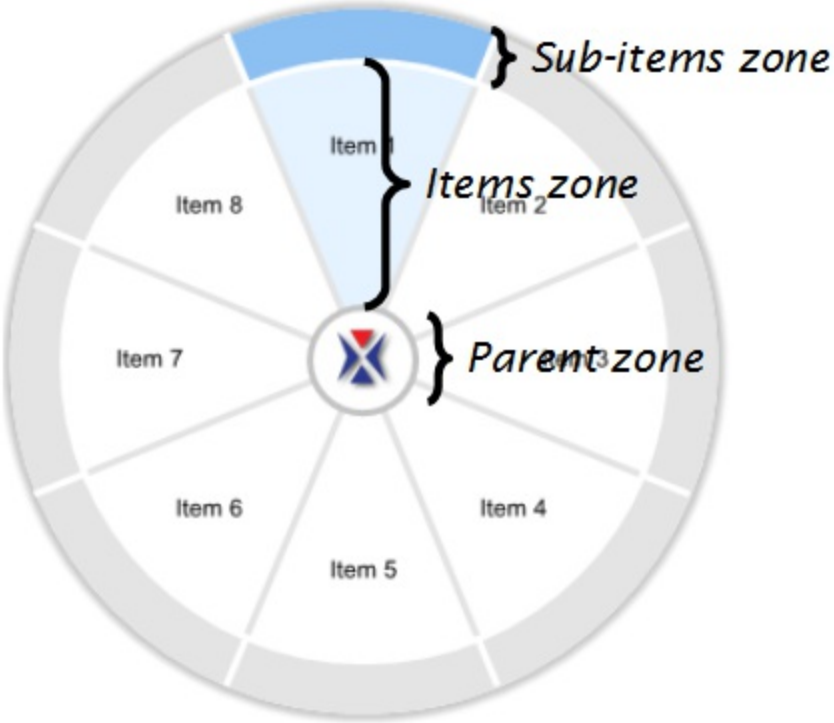
property RadialMenu.SubItemsSize as String

Specifies the size to display the sub-items zone.

Type	Description
String	A String expression that defines the size of the sub-items portion of the control.

By default, The SubItemsSize property is "24*dpi", which indicates 24 pixels if DPI setting is 100%, or 36 pixels (24 * 1.5) if DPI setting is 150%. The SubItemsSize property specifies the size to display the sub-items zone. The ItemsBackColor / ItemsBackAlpha property Specifies the color to show the items portion of the radial menu. The BackColor property specifies the control's background color. The ParentBackColor / ParentBackAlpha property specifies the color / transparency to show the parent portion of the radial menu. The ParentSize property specifies the size to display the parent zone. The SubItemsBackColor / SubItemsBackAlpha property specifies the color to show the sub- items zone of the radial menu. The RadialLineColor(exRadialSubItemsBorder) property specifies the color to show the border around the sub-items section of the control. The RadialLineColor(exRadialSubItemsGridLines) property specifies the color to show the grid lines, inside the sub-items section of the control. The InflateRadialMenu property inflates or deflates the client area of the radial menu control. The InflateItems property Inflates or deflates the client area of the items portion of the control. The InflateCustom property inflates or deflates the client area of the custom portion of the control. The InflateParentPicture property inflates or deflates the client area to display the picture on the background of the parent's zone of the control. The ParentSize property specifies the size to display the parent zone.

The following screen shot, shows the portions/parts/zones of the radial menu:



The ParentSize property supports the following keywords:

- **value**, indicates the radius in pixels, of the radial menu.

The property supports predefined constants, operators and functions as listed bellow:

The constants are (DPI-Aware components):

- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

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 binary range operators, all these with the same priority 5, are :

- **MIN** (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- **MAX** (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

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

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

=: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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 *?* operator 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(%1) = 8* specifies the cells (on the column 1) 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. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54

- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#

Other known operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- 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. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the

trim(" mihai ") returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- **startwith** (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- **endwith** (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- **contains** (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- **left** (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- **right** (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- 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. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"

- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

property RadialMenu.Template as String

Specifies the control's template.

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

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

The Exontrol's eXHelper tool helps you to find easy and quickly the answers and the source code for your questions regarding the usage of our UI components.

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

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

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

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

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

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

For instance, the following script:

```
BeginUpdate
Expanded = True
PicturesPath = "C:\Program Files\Exontrol\ExRadialMenu\Sample\Images"
HTMLPicture("arrow") = "arrow.png"
SubItemsBackColor = RGB(190,190,190)
ShadowColor = SubItemsBackColor
RadialLineColor(6) = SubItemsBackColor
RadialLineColor(5) = -1
Items
{
   Add("Foreground-Color","color_line.png").Items.ToString = "Foreground"
   Add("Background-Color","color_fill.png").Items.ToString = "Background"
   Add("Font","format_font_size_less.png").Items.ToString = "Font"
   Add("Undo","edit_undo.png").Items.ToString = "Undo"
   Add("Redo","edit_redo.png").Items.ToString = "Redo"
   Add("Copy","edit_copy.png").Items.ToString = "Copy"
   Add("List","fileview_text.png").Items.ToString = "List"
   Add("Tag","checkmark_korganizer.png").Items.ToString = "Tag"
}
EndUpdate
```

generates:

property RadialMenu.TemplateDef as Variant

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters)* = *Value*, so our controls provide an alternative using the TemplateDef method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

So, calling the TemplateDef property should be as follows:

```
with (Control)
   TemplateDef = [Dim var_Column]
   TemplateDef = var_Column
   Template = [var_Column.Def(4) = 255]
endwith
```

This sample allocates a variable var_Column, assigns the value to the variable (the second call of the TemplateDef), and the Template call uses the var_Column variable (as an object), to call its Def property with the parameter 4.

Let's say we need to define the background color for a specified column, so we need to call the Def(exCellBackColor) property of the column, to define the color for all cells in the column.

The following **VB6** sample shows setting the Def property such as:

```
With Control
   .Columns.Add("Column 1").Def(exCellBackColor) = 255
   .Columns.Add "Column 2"
   .Items.AddItem 0
   .Items.AddItem 1
```

```
   .Items.AddItem 2
End With
```

In **dBASE Plus**, calling the Def(4) has no effect, instead using the TemplateDef helps you to use properly the Def property as follows:

```
local Control,var_Column

Control = form.Activex1.nativeObject
// Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
with (Control)
   TemplateDef = [Dim var_Column]
   TemplateDef = var_Column
   Template = [var_Column.Def(4) = 255]
endwith
Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The equivalent sample for **XBasic in A5**, is as follows:

```
Dim Control as P
Dim var_Column as P

Control = topparent:CONTROL_ACTIVEX1.activex
' Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
Control.TemplateDef = "Dim var_Column"
Control.TemplateDef = var_Column
Control.Template = "var_Column.Def(4) = 255"

Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The samples just call the Column.Def(4) = Value, using the TemplateDef. The first call of TemplateDef property is "Dim var_Column", which indicates that the next call of the TemplateDef will defines the value of the variable var_Column, in other words, it defines the object var_Column. The last call of the Template property uses the var_Column member to use the x-script and so to set the Def property so a new color is being assigned to the column.

The TemplateDef, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

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

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please

make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

method RadialMenu.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 RadialMenu.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.

By default, the ToolTipDelay property is 500, which indicates that the tooltip is shown after 0.5 seconds. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipFont property to change the tooltip's font. Use the ShowToolTip method to display a custom tooltip. 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. The Tooltip / TooltipTitle property indicates the item's tooltip.

property RadialMenu.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

Type	Description
IFontDisp	A Font object to be used by the control's tooltip.

Use the ToolTipFont property to change the tooltip's font. Use the ShowToolTip method to display a custom tooltip. Use the ToolTipWidth property to specify the width of the tooltip window 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 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 / TooltipTitle property indicates the item's tooltip.

property RadialMenu.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.

By default, the ToolTipPopDelay property is 5000, which indicates that the tooltip remains visible for 5 seconds, while the cursor is not moved. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipFont property to change the tooltip's font. Use the ShowToolTip method to display a custom tooltip. 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. The Tooltip / TooltipTitle property indicates the item's tooltip.

property RadialMenu.ToolTipWidth as Long

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

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

By default, the ToolTipWidth property is 196 pixels. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ShowToolTip method to display a custom tooltip. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. The Tooltip / TooltipTitle property indicates the item's tooltip.

property RadialMenu.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

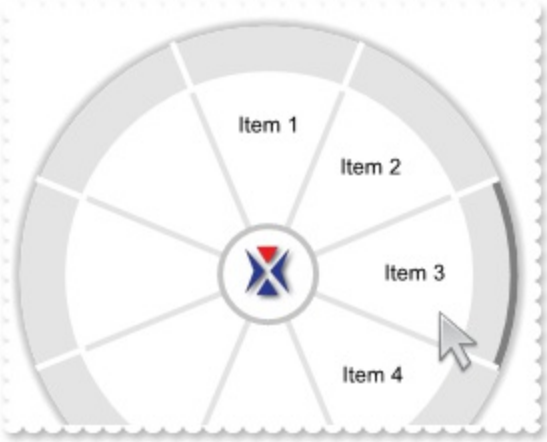
The ToString property loads or saves the control items from a string. The ToString property is equivalent with Root.Items.ToString property.

The user can add new items to the control using any of the following:

- Add method, adds a new item to the control. The Add method can be used to add child-items as well.
- ToString property of the Items collection, loads or saves the Items collection using string representation.
- ToString property of the control, loads or saves the Items collection using string representation.

The Remove method removes a specified item. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item (child, radial-slider or gauge). The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

For instance, the *"Item 1,Item 2,Item 3, Item 4"*, generates the following screen shot:



For instance, the *"Item 1,Item 2[scap=<font ;6>continue],Item 3 (Child 1,Child 2)"*, 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> ::= "scap" | "img" | "simg" | "bg" | "sbg" | "bga" | "sbga" | "fg" | "sfg" | "ttp" | "sttp" | "ttpt" | "sttpt" | "data" | "sdata" | "browse" | "custom" | "value"

where the <CAPTION> is the HTML caption to be shown on the item, equivalent with Caption(exRadialItems). The <VALUE> indicates the value of giving property.

- 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. This option is equivalent with the BackColor(exRadialItems) property.
- sbg=<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. This option is equivalent with the BackColor(exRadialSubItems) property.
- bga=<VALUE>, specifies the value of alpha / opacity channel to show the item's background color, where <VALUE> is a BYTE value. This option is equivalent with the BackAlpha(exRadialItems) property.
- sbga=<VALUE>, specifies the value of alpha / opacity channel to show the item's background color, where <VALUE> is a BYTE value. This option is equivalent with the BackAlpha(exRadialSubItems) property.
- browse=<VALUE>, specifies what the item displays, when the user clicks/browses it, where <VALUE> is a BrowseItemEnum value (0, 1 or 2). This option is equivalent with the BrowseType property.
- custom=<VALUE>, indicates the custom object to be shown when the user clicks/browses the item, where <VALUE> is a RadialCustomTypeEnum value (0, 16 or 32). This option is equivalent with the BrowseCustomType property.
- data=<VALUE>, indicates the item's user data, where <VALUE> is any expression.

This option is equivalent with the UserData(exRadialItems) property.

- sdata=<VALUE>, indicates the item's user data, where <VALUE> is any expression. This option is equivalent with the UserData(exRadialSubItems) property.
- fg=<VALUE>, indicates the item's foreground color, where <VALUE> is 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). This option is equivalent with the ForeColor(exRadialItems) property.
- sfg=<VALUE>, indicates the item's foreground color, where <VALUE> is 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). This option is equivalent with the ForeColor(exRadialSubItems) property.
- img=<VALUE>, indicates the item's image, where <VALUE> is name of the picture, key, icon, and so on. This option is equivalent with the Image(exRadialItems) property.
- simg=<VALUE>, indicates the item's image, where <VALUE> is name of the picture, key, icon, and so on. This option is equivalent with the Image(exRadialSubItems) property.
- scap=<VALUE>, indicates the item's caption, where <VALUE> is HTML text to be shown on the sub-item zone of the item. This option is equivalent with the Caption(exRadialSubItems) property.
- ttp=<VALUE>, indicates the item's tooltip, where <VALUE> is HTML text to be shown when the cursor hovers the item. This option is equivalent with the Tooltip(exRadialItems) property.
- sttp=<VALUE>, indicates the item's tooltip, where <VALUE> is HTML text to be shown when the cursor hovers the item. This option is equivalent with the Tooltip(exRadialSubItems) property.
- ttpt=<VALUE>, indicates the title of the item's tooltip, where <VALUE> is the title of the item's tooltip. This option is equivalent with the TooltipTitle(exRadialItems) property.
- sttpt=<VALUE>, indicates the title of the item's tooltip, where <VALUE> is the title of the item's tooltip. This option is equivalent with the TooltipTitle(exRadialSubItems) property.
- value=<VALUE>, indicates the item's value, where <VALUE> is the value. This option is equivalent with the BrowseCustom(exRadialCustomSliderValue) property.

property RadialMenu.ToTemplate ([DefaultTemplate as Variant]) as String

Generates the control's template.

Type	Description
DefaultTemplate as Variant	A String expression that indicates the default format used to define the control's template at runtime, or a string expression that indicates the path to the file being used to define the default template (like c:\temp\teml.bin). If it is missing (by default), the control's uses the default implementation (listed bellow) to define the control's template, at runtime. Each line in the DefaultTemplate parameter, defines a property or an instruction to generate the template.
String	A String expression that indicates the control's template.

Only for future use. Use the ToTemplate property to save the control's content to a template string. The ToTemplate property saves the control's properties based on the default template. Use the ToTemplate property to copy the control's content to another instance. The ToTemplate property can save pictures, icons, binary arrays, objects, collections, and so on based on the DefaultTemplate parameter.

The DefaultTemplate parameter indicates the format of the template being used to generate the control's template at runtime. If the DefaultTemplate parameter is missing, the control's uses its default template listed bellow. The DefaultTemplate parameter defines the list of properties and instructions that generates the control's template. Remove the properties and objects, in the default template, that you don't need in the generated template script. Use the Template property to apply the template to the control. Use the Template property to execute code by passing instructions as a string (template string). The Template script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters. The Template format contains a list of instructions that loads data and change properties for the objects in the control. Use the AllowCopyTemplate property to copy the control's content to the clipboard, in template format, using the the Shift + Ctrl + Alt + Insert sequence.

property RadialMenu.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 RadialMenu.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.

ExRadialMenu events

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {1604BDE1-D48F-4D3F-B51B-49C0CD74236C}. The object's program identifier is: "Exontrol.RadialMenu". The /COM object module is: "ExRadialMenu.dll"

The eXRadialMenu (radial or pie menu) component is similar to the Microsoft's OneNote radial menu with ability to customize the appearance and functionality. The component is designed using tree structure so an item can hold none or more children, and so any item can be browsed, and show its children around it. An item can display a collection of child items, as well as a radial slider, or any other gauge / knob control. The eXRadialMenu is written from scratch, and does not depend on Windows 7, 8, 10 and so requires no dependencies to any other third party library.

The RadialMenu component supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
BrowseItem	Notifies once the user browses for a new item.
ChangeItem	Occurs when the user changes the item's value.
Click	Occurs when the user presses and then releases the left mouse button over the control.
DblClick	Occurs when the user dblclk the left mouse button over an object.
Event	Notifies the application once the control fires an event.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
MouseWheel	Occurs when the mouse wheel moves while the control has focus
RClick	Occurs once the user right clicks the control.
SelectItem	Notifies once the user selects an item.
SelectParent	Occurs once the user clicks the parent of the item.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

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

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

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

C#

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

VB

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

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

C#

```
private void AnchorClick(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_AnchorClickEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure AnchorClick(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_AnchorClickEvent);
begin
end;
```

Powe…

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

end event AnchorClick
```

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS AnchorID,Options

```

Xbas…

```
PROCEDURE OnAnchorClick(oRadialMenu,AnchorID,Options)
```

```
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComAnchorClick String   llAnchorID String   llOptions
    Forward Send OnComAnchorClick llAnchorID llOptions
End_Procedure
```

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event BrowseItem (Item as Item)

Notifies once the user browses for a new item.

Type	Description
Item as Item	An Item object being browsed.

The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The BrowseItem property specifies the item currently browsed. The AllowBrowseItem property specifies that the a new item gets browsed once the user clicks item. The SelectItem event notifies once the user selects an item. The SelectParent event occurs once the user clicks the parent of the item.

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

C#
```
private void BrowseItem(object sender,exontrol.EXRADIALMENULib.Item   Item)
{
}
```

VB
```
Private Sub BrowseItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXRADIALMENULib.Item) Handles BrowseItem
End Sub
```

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

C#
```
private void BrowseItem(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_BrowseItemEvent e)
{
}
```

C++
```
void OnBrowseItem(LPDISPATCH   Item)
{
}
```

C++ Builder
```
void __fastcall BrowseItem(TObject *Sender,Exradialmenulib_tlb::IItem   *Item)
{
}
```

Delphi
```
procedure BrowseItem(ASender: TObject; Item : IItem);
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure BrowseItem(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_BrowseItemEvent);
begin
end;
```

Powe…

```
begin event BrowseItem(oleobject  Item)


end event BrowseItem
```

VB.NET

```
Private Sub BrowseItem(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_BrowseItemEvent) Handles
BrowseItem
End Sub
```

VB6

```
Private Sub BrowseItem(ByVal Item As EXRADIALMENULibCtl.IItem)
End Sub
```

VBA

```
Private Sub BrowseItem(ByVal Item As Object)
End Sub
```

VFP

```
LPARAMETERS Item

```

Xbas…

```
PROCEDURE OnBrowseItem(oRadialMenu,Item)


RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComBrowseItem Variant   llItem
    Forward Send OnComBrowseItem llItem
End_Procedure
```

Visual Objects

```
METHOD OCX_BrowseItem(Item) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_BrowseItem(COM   _Item)
{
}
```

XBasic

```
function BrowseItem as v (Item  as  OLE::Exontrol.RadialMenu.1::IItem)
end function
```

dBASE

```
function nativeObject_BrowseItem(Item)
return
```

event ChangeItem (Item as Item)

Occurs when the user changes the item's value.

Type	Description
Item as Item	An Item object whose value has been changed.

The ChangeItem event occurs when the user changes the item's value. For instance, the ChangeItem event occurs once the user change the value of a radial slider. The BrowseType property specifies what the item displays, when the user clicks/browses it. The BrowseCustomType property indicates the custom object to be shown when the user clicks/browses the item. The BrowseCustom property gets or sets a value for specified property, when browsing custom control.

The control can display:

- child items (BrowseType property is exBrowseItemChild)
- radial-slider (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomSlider)
- gauge control (BrowseType property is exBrowseItemCustom, and BrowseCustomType property is exRadialCustomGauge). The control displays/edit data using the using the Exontrol's ExGauge component.

Currently, the ChangeItem event notifies when the user changes:

- BrowseCustom(exRadialCustomSliderValue) property, gets or sets the radial slider's value (double expression).

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

C#
```
private void ChangeItem(object sender,exontrol.EXRADIALMENULib.Item   Item)
{
}
```

VB
```
Private Sub ChangeItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXRADIALMENULib.Item) Handles ChangeItem
End Sub
```

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

C#
```
private void ChangeItem(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_ChangeItemEvent e)
{
```

```
}
```

C++

```
void OnChangeItem(LPDISPATCH  Item)
{
}
```

C++ Builder

```
void __fastcall ChangeItem(TObject *Sender,Exradialmenulib_tlb::IItem  *Item)
{
}
```

Delphi

```
procedure ChangeItem(ASender: TObject; Item : IItem);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ChangeItem(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_ChangeItemEvent);
begin
end;
```

Powe…

```
begin event ChangeItem(oleobject  Item)

end event ChangeItem
```

VB.NET

```
Private Sub ChangeItem(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_ChangeItemEvent) Handles
ChangeItem
End Sub
```

VB6

```
Private Sub ChangeItem(ByVal Item As EXRADIALMENULibCtl.IItem)
End Sub
```

VBA

```
Private Sub ChangeItem(ByVal Item As Object)
End Sub
```

VFP

```
LPARAMETERS Item

```

Xbas…

```
PROCEDURE OnChangeItem(oRadialMenu,Item)
```

```
RETURN
```

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

Java…
```
<SCRIPT EVENT="ChangeItem(Item)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ChangeItem(Item)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComChangeItem Variant   llItem
   Forward Send OnComChangeItem llItem
End_Procedure
```

Visual Objects
```
METHOD OCX_ChangeItem(Item) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ChangeItem(COM   _Item)
{
}
```

XBasic
```
function ChangeItem as v (Item  as  OLE::Exontrol.RadialMenu.1::IItem)
end function
```

dBASE
```
function nativeObject_ChangeItem(Item)
return
```

event Click ()

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

Type	Description

The Click event is fired when the user releases the left mouse button over the control. The Click event is not fired if you click, drag and release the mouse over the control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. The BrowseItem property specifies the item currently browsed. The SelectItem event notifies once the user selects an item. The SelectParent event occurs once the user clicks the parent of the item. The ItemFromPoint property gets the item from the cursor. The IndexFromPoint property gets the item from the cursor. The AnchorFromPoint property retrieves the identifier of the anchor from point.

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

C#

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

VB

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

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

C#

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

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event Click()


end event Click
```

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnClick(oRadialMenu)

RETURN
```

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

Java…

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

VBSc…

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

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComClick
    Forward Send OnComClick
End_Procedure
```

Visual Objects

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

X++

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

XBasic

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

dBASE

```
function nativeObject_Click()
return
```

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

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

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

The DblClick event is fired when user double clicks the control. The Click event is not fired if you click, drag and release the mouse over the control. The ItemFromPoint property gets the item from the cursor. The IndexFromPoint property gets the item from the cursor. The AnchorFromPoint property retrieves the identifier of the anchor from point. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released.

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

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

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

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

C#
```
private void DblClick(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_DblClickEvent e)
{
}
```

C++
```
void OnDblClick(short   Shift,long   X,long   Y)
{
```

```
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure DblClick(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_DblClickEvent);
begin
end;
```

Powe…

```
begin event DblClick(integer  Shift,long  X,long  Y)

end event DblClick
```

VB.NET

```
Private Sub DblClick(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_DblClickEvent) Handles DblClick
End Sub
```

VB6

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

VBA

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

VFP

```
LPARAMETERS Shift,X,Y

```

Xbas…

```
PROCEDURE OnDblClick(oRadialMenu,Shift,X,Y)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event Event (EventID as Long)

Notifies the application once the control fires an event.

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

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

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

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

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

Here's how the output is shown, when printing the EventParam(-2) during the Event event:

```
BrowseItem/11( [Object] )
MouseMove/-606( 0 , 0 , 15 , 139 )
MouseDown/-605( 1 , 0 , 165 , 496 )
ChangeItem/13( [Object] )
MouseMove/-606( 1 , 0 , 162 , 488 )
ChangeItem/13( [Object] )
MouseMove/-606( 1 , 0 , 160 , 470 )
ChangeItem/13( [Object] )
MouseMove/-606( 1 , 0 , 157 , 434 )
ChangeItem/13( [Object] )
MouseUp/-607( 1 , 0 , 157 , 434 )
```

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,
AxEXRADIALMENULib._IRadialMenuEvents_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:
AxEXRADIALMENULib._IRadialMenuEvents_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
AxEXRADIALMENULib._IRadialMenuEvents_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(oRadialMenu,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 KeyDown (ByRef KeyCode as Integer, Shift as Integer)

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

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

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

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

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

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

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

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

C#
```
private void KeyDownEvent(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_KeyDownEvent e)
```

```
{
}
```

C++

```
void OnKeyDown(short FAR*   KeyCode,short   Shift)
{
}
```

C++ Builder

```
void __fastcall KeyDown(TObject *Sender,short *   KeyCode,short   Shift)
{
}
```

Delphi

```
procedure KeyDown(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyDownEvent(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_KeyDownEvent);
begin
end;
```

Powe…

```
begin event KeyDown(integer  KeyCode,integer  Shift)

end event KeyDown
```

VB.NET

```
Private Sub KeyDownEvent(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_KeyDownEvent) Handles
KeyDownEvent
End Sub
```

VB6

```
Private Sub KeyDown(KeyCode As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyDown(KeyCode As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift

```

Xbas…

```
PROCEDURE OnKeyDown(oRadialMenu,KeyCode,Shift)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event KeyPress (ByRef KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	(By Reference) An integer that returns a standard numeric ANSI keycode.

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

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

C#

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

VB

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

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

C#

```
private void KeyPressEvent(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_KeyPressEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure KeyPressEvent(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_KeyPressEvent);
begin
end;
```

Powe…

```
begin event KeyPress(integer  KeyAscii)

end event KeyPress
```

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS KeyAscii

```

Xbas…

```
PROCEDURE OnKeyPress(oRadialMenu,KeyAscii)

RETURN
```

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

Java…

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

VBSc…

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

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyPress Short   llKeyAscii
    Forward Send OnComKeyPress llKeyAscii
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyPress(COMVariant /*short*/   _KeyAscii)
{
}
```

XBasic

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

dBASE

```
function nativeObject_KeyPress(KeyAscii)
return
```

event KeyUp (ByRef KeyCode as Integer, Shift as Integer)

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

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

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

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

C#

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

VB

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

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

C#

```
private void KeyUpEvent(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_KeyUpEvent e)
{
}
```

C++

```
void OnKeyUp(short FAR*   KeyCode,short   Shift)
{
}
```

C++ Builder

```
void __fastcall KeyUp(TObject *Sender,short *   KeyCode,short   Shift)
{
```

```
}
```

Delphi

```
procedure KeyUp(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyUpEvent(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_KeyUpEvent);
begin
end;
```

Powe…

```
begin event KeyUp(integer  KeyCode,integer  Shift)

end event KeyUp
```

VB.NET

```
Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_KeyUpEvent) Handles KeyUpEvent
End Sub
```

VB6

```
Private Sub KeyUp(KeyCode As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyUp(KeyCode As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift

```

Xbas…

```
PROCEDURE OnKeyUp(oRadialMenu,KeyCode,Shift)

RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
```

```
Function KeyUp(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

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

Occurs when the user presses a mouse button.

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

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

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

C#

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

VB

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

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

C#

```
private void MouseDownEvent(object sender,
```

```
AxEXRADIALMENULib._IRadialMenuEvents_MouseDownEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure MouseDownEvent(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_MouseDownEvent);
begin
end;
```

Powe…

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

end event MouseDown
```

VB.NET

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

VB6

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

VBA

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

VFP

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

Xbas…

```
PROCEDURE OnMouseDown(oRadialMenu,Button,Shift,X,Y)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

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

Occurs when the user moves the mouse.

Type	Description
Button as Integer	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down. Gets which mouse button was pressed as 1 for Left Mouse Button, 2 for Right Mouse Button and 4 for Middle Mouse Button.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. The ItemFromPoint property gets the item from the cursor. The IndexFromPoint property gets the item from the cursor. The ParentOnPoint property indicates if the point hits the parent zone of the radial menu. The AnchorFromPoint property retrieves the identifier of the anchor from point.

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

C#

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

VB

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

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

C#

```
private void MouseMoveEvent(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_MouseMoveEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure MouseMoveEvent(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_MouseMoveEvent);
begin
end;
```

Powe…

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


end event MouseMove
```

VB.NET

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

VB6

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

VBA

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

VFP

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

Xbas…

```
PROCEDURE OnMouseMove(oRadialMenu,Button,Shift,X,Y)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

```
end function
```

dBASE

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

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

Occurs when the user releases a mouse button.

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

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

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

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

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

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

C#
```
private void MouseUpEvent(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_MouseUpEvent e)
```

```
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure MouseUpEvent(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_MouseUpEvent);
begin
end;
```

Powe…

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

end event MouseUp
```

VB.NET

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

VB6

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

VBA

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

VFP

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

Xbas…
```
PROCEDURE OnMouseUp(oRadialMenu,Button,Shift,X,Y)

RETURN
```

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

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

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

Visual Data…
```
Procedure OnComMouseUp Short   llButton Short   llShift OLE_XPOS_PIXELS   llX
OLE_YPOS_PIXELS   llY
   Forward Send OnComMouseUp llButton llShift llX llY
End_Procedure
```

Visual Objects
```
METHOD OCX_MouseUp(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_MouseUp(int   _Button,int   _Shift,int   _X,int   _Y)
{
}
```

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

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

event MouseWheel (Delta as Long)

Occurs when the mouse wheel moves while the control has focus

Type	Description
Delta as Long	A long expression that specifies the direction and the quantity that the mouse wheel has been rolled. For instance, 1 indicates that the user rolls the mouse wheel up, -1 indicates that the user rolls the mouse wheel down. Any other value may indicate that the mouse wheel has been rolled quicker.

The MouseWheel occurs when the mouse wheel is rolled. You can use the MouseWheel event to perform different actions on any layer when the user rolls the mouse wheel. The ItemFromPoint property gets the item from the cursor. The IndexFromPoint property gets the item from the cursor. The AnchorFromPoint property retrieves the identifier of the anchor from point. The FormatABC method formats the A,B,C values based on the giving expression and returns the result.

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

C#

```
private void MouseWheel(object sender,int   Delta)
{
}
```

VB

```
Private Sub MouseWheel(ByVal sender As System.Object,ByVal Delta As Integer)
Handles MouseWheel
End Sub
```

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

C#

```
private void MouseWheel(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_MouseWheelEvent e)
{
}
```

C++

```
void OnMouseWheel(long   Delta)
{
}
```

C++ Builder

```
void __fastcall MouseWheel(TObject *Sender,long   Delta)
{
}
```

Delphi

```
procedure MouseWheel(ASender: TObject; Delta : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseWheel(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_MouseWheelEvent);
begin
end;
```

Powe…

```
begin event MouseWheel(long  Delta)


end event MouseWheel
```

VB.NET

```
Private Sub MouseWheel(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_MouseWheelEvent) Handles
MouseWheel
End Sub
```

VB6

```
Private Sub MouseWheel(ByVal Delta As Long)
End Sub
```

VBA

```
Private Sub MouseWheel(ByVal Delta As Long)
End Sub
```

VFP

```
LPARAMETERS Delta


```

Xbas…

```
PROCEDURE OnMouseWheel(oRadialMenu,Delta)


RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComMouseWheel Integer   llDelta
    Forward Send OnComMouseWheel llDelta
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseWheel(Delta) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseWheel(int   _Delta)
{
}
```

XBasic

```
function MouseWheel as v (Delta  as  N)
end function
```

dBASE

```
function nativeObject_MouseWheel(Delta)
return
```

event RClick ()

Occurs once the user right clicks the control.

Type	Description

Use the RClick event to add your context menu. The RClick event notifies your application when the user right clicks the control. Use the Click event to notify your application that the user clicks the control (using the left mouse button). Use the MouseDown or MouseUp event if you require the cursor position during the RClick event. The ItemFromPoint property gets the item from the cursor. The IndexFromPoint property gets the item from the cursor. The AnchorFromPoint property retrieves the identifier of the anchor from point.

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

C#
```
private void RClick(object sender)
{
}
```

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

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

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

C++
```
void OnRClick()
{
}
```

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

Delphi
```
procedure RClick(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

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

Powe…

```
begin event RClick()

end event RClick
```

VB.NET

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

VB6

```
Private Sub RClick()
End Sub
```

VBA

```
Private Sub RClick()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnRClick(oRadialMenu)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComRClick
    Forward Send OnComRClick
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_RClick()
{
}
```

XBasic

```
function RClick as v ()
end function
```

dBASE

```
function nativeObject_RClick()
return
```

event SelectItem (Item as Item)

Notifies once the user selects an item.

Type	Description
Item as Item	The Item being clicked / selected.

The SelectItem event notifies once the user selects an item. The SelectItem event is fired when user clicks an item with no child items. The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The SelectParent event occurs once the user clicks the parent of the item. The SelectedIndex property gets or sets a value that indicates index to be selected.

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

C#
```
private void SelectItem(object sender,exontrol.EXRADIALMENULib.Item  Item)
{
}
```

VB
```
Private Sub SelectItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXRADIALMENULib.Item) Handles SelectItem
End Sub
```

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

C#
```
private void SelectItem(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent e)
{
}
```

C++
```
void OnSelectItem(LPDISPATCH  Item)
{
}
```

C++ Builder
```
void __fastcall SelectItem(TObject *Sender,Exradialmenulib_tlb::IItem  *Item)
{
}
```

Delphi
```
procedure SelectItem(ASender: TObject; Item : IItem);
```

```
begin
end;
```

Delphi 8 (.NET only)

```
procedure SelectItem(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent);
begin
end;
```

Powe…

```
begin event SelectItem(oleobject  Item)

end event SelectItem
```

VB.NET

```
Private Sub SelectItem(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent) Handles SelectItem
End Sub
```

VB6

```
Private Sub SelectItem(ByVal Item As EXRADIALMENULibCtl.IItem)
End Sub
```

VBA

```
Private Sub SelectItem(ByVal Item As Object)
End Sub
```

VFP

```
LPARAMETERS Item

```

Xbas…

```
PROCEDURE OnSelectItem(oRadialMenu,Item)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComSelectItem Variant   llItem
    Forward Send OnComSelectItem llItem
End_Procedure
```

Visual Objects

```
METHOD OCX_SelectItem(Item) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_SelectItem(COM   _Item)
{
}
```

XBasic

```
function SelectItem as v (Item  as  OLE::Exontrol.RadialMenu.1::IItem)
end function
```

dBASE

```
function nativeObject_SelectItem(Item)
return
```

The following sample shows how you can select an item, once the user clicks it:

VBA (MS Access, Excell...)

```
' SelectItem event - Notifies once the user selects an item.
Private Sub RadialMenu1_SelectItem(ByVal Item As Object)
    ' SelectedIndex(3) = Item.Index
End Sub

With RadialMenu1
    .BeginUpdate
    .SelBackAlpha(1) = 32
    .SelBackAlpha(2) = 128
    .SelForeColor(3) = RGB(0,0,0)
    .RadialLineSize(8) = -1
    .RadialLineAlpha(8) = 32
    .RadialLineColor(11) = -1
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .SelectedIndex(3) = 0
```

```
    .EndUpdate
End With
```

VB6

```
' SelectItem event - Notifies once the user selects an item.
Private Sub RadialMenu1_SelectItem(ByVal Item As EXRADIALMENULibCtl.IItem)
    ' SelectedIndex(3) = Item.Index
End Sub

With RadialMenu1
    .BeginUpdate
    .SelBackAlpha(exRadialItems) = 32
    .SelBackAlpha(exRadialSubItems) = 128
    .SelForeColor(exRadialFullItems) = RGB(0,0,0)
    .RadialLineSize(exRadialHotParent) = -1
    .RadialLineAlpha(exRadialHotParent) = 32
    .RadialLineColor(exRadialHotFullItem) = -1
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .SelectedIndex(exRadialFullItems) = 0
    .EndUpdate
End With
```

VB.NET

```
' SelectItem event - Notifies once the user selects an item.
Private Sub Exradialmenu1_SelectItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXRADIALMENULib.Item) Handles Exradialmenu1.SelectItem
    ' SelectedIndex(3) = Item.Index
End Sub

With Exradialmenu1
    .BeginUpdate()
    .set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)

.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128
```

```
.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,Colo

.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)

.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotParent,3

.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exRadialHotFullIte

    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"

.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)

    .EndUpdate()
End With
```

VB.NET for /COM

```
' SelectItem event - Notifies once the user selects an item.
Private Sub AxRadialMenu1_SelectItem(ByVal sender As System.Object, ByVal e As
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent) Handles
AxRadialMenu1.SelectItem
    ' SelectedIndex(3) = Item.Index
End Sub

With AxRadialMenu1
    .BeginUpdate()
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32)
    .set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128)
    .set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)
    .set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1)
    .set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32)
    .set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,-1)
```

```
    .Expanded = True
    .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
    .set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0)
    .EndUpdate()
End With
```

C++

```
// SelectItem event - Notifies once the user selects an item.
void OnSelectItemRadialMenu1(LPDISPATCH  Item)
{
   // SelectedIndex(3) = Item.Index
}

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

    #import <ExRadialMenu.dll>
    using namespace EXRADIALMENULib;
*/
EXRADIALMENULib::IRadialMenuPtr spRadialMenu1 =
GetDlgItem(IDC_RADIALMENU1)->GetControlUnknown();
spRadialMenu1->BeginUpdate();
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialItems,32);
spRadialMenu1->PutSelBackAlpha(EXRADIALMENULib::exRadialSubItems,128);
spRadialMenu1->PutSelForeColor(EXRADIALMENULib::exRadialFullItems,RGB(0,0,0));
spRadialMenu1->PutRadialLineSize(EXRADIALMENULib::exRadialHotParent,-1);
spRadialMenu1->PutRadialLineAlpha(EXRADIALMENULib::exRadialHotParent,32);
spRadialMenu1->PutRadialLineColor(EXRADIALMENULib::exRadialHotFullItem,-1);
spRadialMenu1->PutExpanded(VARIANT_TRUE);
spRadialMenu1->GetItems()->PutToString(L"Item 1,Item 2,Item 3,Item 4,Item 5,Item
6,Item 7,Item 8");
spRadialMenu1->PutSelectedIndex(EXRADIALMENULib::exRadialFullItems,0);
spRadialMenu1->EndUpdate();
```

C++ Builder

```
// SelectItem event - Notifies once the user selects an item.
void __fastcall TForm1::RadialMenu1SelectItem(TObject
*Sender,Exradialmenulib_tlb::IItem  *Item)
{
  // SelectedIndex(3) = Item.Index
}

RadialMenu1->BeginUpdate();
RadialMenu1->SelBackAlpha[Exradialmenulib_tlb::RadialItemsEnum::exRadialItems] =
32;
RadialMenu1-
>SelBackAlpha[Exradialmenulib_tlb::RadialItemsEnum::exRadialSubItems] = 128;
RadialMenu1-
>SelForeColor[Exradialmenulib_tlb::RadialItemsEnum::exRadialFullItems] =
RGB(0,0,0);
RadialMenu1-
>RadialLineSize[Exradialmenulib_tlb::RadialLineEnum::exRadialHotParent] = -1;
RadialMenu1-
>RadialLineAlpha[Exradialmenulib_tlb::RadialLineEnum::exRadialHotParent] = 32;
RadialMenu1-
>RadialLineColor[Exradialmenulib_tlb::RadialLineEnum::exRadialHotFullItem] = -1;
RadialMenu1->Expanded = true;
RadialMenu1->Items->ToString = L"Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
RadialMenu1-
>SelectedIndex[Exradialmenulib_tlb::RadialItemsEnum::exRadialFullItems] = 0;
RadialMenu1->EndUpdate();
```

C#

```
// SelectItem event - Notifies once the user selects an item.
private void exradialmenu1_SelectItem(object
sender,exontrol.EXRADIALMENULib.Item  Item)
{
  // SelectedIndex(3) = Item.Index
```

```
}
//this.exradialmenu1.SelectItem += new
exontrol.EXRADIALMENULib.exg2antt.SelectItemEventHandler(this.exradialmenu1_Selec


exradialmenu1.BeginUpdate();
exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.set_SelBackAlpha(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadi

exradialmenu1.set_SelForeColor(exontrol.EXRADIALMENULib.RadialItemsEnum.exRadia

exradialmenu1.set_RadialLineSize(exontrol.EXRADIALMENULib.RadialLineEnum.exRadia

exradialmenu1.set_RadialLineAlpha(exontrol.EXRADIALMENULib.RadialLineEnum.exRad

exradialmenu1.set_RadialLineColor32(exontrol.EXRADIALMENULib.RadialLineEnum.exR

exradialmenu1.Expanded = true;
exradialmenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8";
exradialmenu1.set_SelectedIndex(exontrol.EXRADIALMENULib.RadialItemsEnum.exRa

exradialmenu1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload="Init()">
<SCRIPT FOR="RadialMenu1" EVENT="SelectItem(Item)" LANGUAGE="JScript">
  // SelectedIndex(3) = Item.Index
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
```

```
function Init()
{
   RadialMenu1.BeginUpdate();
   RadialMenu1.SelBackAlpha(1) = 32;
   RadialMenu1.SelBackAlpha(2) = 128;
   RadialMenu1.SelForeColor(3) = 0;
   RadialMenu1.RadialLineSize(8) = -1;
   RadialMenu1.RadialLineAlpha(8) = 32;
   RadialMenu1.RadialLineColor(11) = -1;
   RadialMenu1.Expanded = true;
   RadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
   RadialMenu1.SelectedIndex(3) = 0;
   RadialMenu1.EndUpdate();
}
</SCRIPT>
</BODY>
```

VBScript

```
<BODY onload="Init()">
<SCRIPT LANGUAGE="VBScript">
Function RadialMenu1_SelectItem(Item)
   ' SelectedIndex(3) = Item.Index
End Function
</SCRIPT>

<OBJECT CLASSID="clsid:1604BDE1-D48F-4D3F-B51B-49C0CD74236C"
id="RadialMenu1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With RadialMenu1
      .BeginUpdate
      .SelBackAlpha(1) = 32
      .SelBackAlpha(2) = 128
```

```
      .SelForeColor(3) = RGB(0,0,0)
      .RadialLineSize(8) = -1
      .RadialLineAlpha(8) = 32
      .RadialLineColor(11) = -1
      .Expanded = True
      .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
      .SelectedIndex(3) = 0
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
// SelectItem event - Notifies once the user selects an item.
private void axRadialMenu1_SelectItem(object sender,
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent e)
{
   // SelectedIndex(3) = Item.Index
}
//this.axRadialMenu1.SelectItem += new
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEventHandler(this.axRadialMenu1_


axRadialMenu1.BeginUpdate();
axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,3

axRadialMenu1.set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubIte

axRadialMenu1.set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullIter
(uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0)));
axRadialMenu1.set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotPar

axRadialMenu1.set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotP
```

```
axRadialMenu1.set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFu

axRadialMenu1.Expanded = true;
axRadialMenu1.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8";
axRadialMenu1.set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullI

axRadialMenu1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
// SelectItem event - Notifies once the user selects an item.
void onEvent_SelectItem(COM   _Item)
{
   // SelectedIndex(3) = Item.Index
   ;
}

public void init()
{
   ;

   super();

   exradialmenu1.BeginUpdate();
   exradialmenu1.SelBackAlpha(1/*exRadialItems*/,32);
   exradialmenu1.SelBackAlpha(2/*exRadialSubItems*/,128);
   exradialmenu1.SelForeColor(3/*exRadialFullItems*/,WinApi::RGB2int(0,0,0));
   exradialmenu1.RadialLineSize(8/*exRadialHotParent*/,-1);
   exradialmenu1.RadialLineAlpha(8/*exRadialHotParent*/,32);
   exradialmenu1.RadialLineColor(11/*exRadialHotFullItem*/,-1);
   exradialmenu1.Expanded(true);
   exradialmenu1.Items().ToString("Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8");
   exradialmenu1.SelectedIndex(3/*exRadialFullItems*/,0);
   exradialmenu1.EndUpdate();
```

```
}
```

Delphi 8 (.NET only)

```
// SelectItem event - Notifies once the user selects an item.
procedure TWinForm1.AxRadialMenu1_SelectItem(sender: System.Object; e:
AxEXRADIALMENULib._IRadialMenuEvents_SelectItemEvent);
begin
   // SelectedIndex(3) = Item.Index
end;

with AxRadialMenu1 do
begin
   BeginUpdate();
   set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialItems,32);
   set_SelBackAlpha(EXRADIALMENULib.RadialItemsEnum.exRadialSubItems,128);
   set_SelForeColor(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,$0);
   set_RadialLineSize(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,-1);
   set_RadialLineAlpha(EXRADIALMENULib.RadialLineEnum.exRadialHotParent,32);

set_RadialLineColor(EXRADIALMENULib.RadialLineEnum.exRadialHotFullItem,$ffffffff);

   Expanded := True;
   Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
   set_SelectedIndex(EXRADIALMENULib.RadialItemsEnum.exRadialFullItems,0);
   EndUpdate();
end
```

Delphi (standard)

```
// SelectItem event - Notifies once the user selects an item.
procedure TForm1.RadialMenu1SelectItem(ASender: TObject; Item : IItem);
begin
   // SelectedIndex(3) = Item.Index
end;

with RadialMenu1 do
begin
```

```
  BeginUpdate();
  SelBackAlpha[EXRADIALMENULib_TLB.exRadialItems] := 32;
  SelBackAlpha[EXRADIALMENULib_TLB.exRadialSubItems] := 128;
  SelForeColor[EXRADIALMENULib_TLB.exRadialFullItems] := $0;
  RadialLineSize[EXRADIALMENULib_TLB.exRadialHotParent] := -1;
  RadialLineAlpha[EXRADIALMENULib_TLB.exRadialHotParent] := 32;
  RadialLineColor[EXRADIALMENULib_TLB.exRadialHotFullItem] := $ffffffff;
  Expanded := True;
  Items.ToString := 'Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8';
  SelectedIndex[EXRADIALMENULib_TLB.exRadialFullItems] := 0;
  EndUpdate();
end
```

VFP

```
*** SelectItem event - Notifies once the user selects an item. ***
LPARAMETERS Item
   *** SelectedIndex(3) = Item.Index

with thisform.RadialMenu1
   .BeginUpdate
   .Object.SelBackAlpha(1) = 32
   .Object.SelBackAlpha(2) = 128
   .Object.SelForeColor(3) = RGB(0,0,0)
   .Object.RadialLineSize(8) = -1
   .Object.RadialLineAlpha(8) = 32
   .Object.RadialLineColor(11) = -1
   .Expanded = .T.
   .Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
   .Object.SelectedIndex(3) = 0
   .EndUpdate
endwith
```

dBASE Plus

```
/*
with (this.EXRADIALMENUACTIVEXCONTROL1.nativeObject)
   SelectItem = class::nativeObject_SelectItem
```

```
endwith
*/
// Notifies once the user selects an item.
function nativeObject_SelectItem(Item)
   /* SelectedIndex(3) = Item.Index */
   oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
return

local oRadialMenu

oRadialMenu = form.EXRADIALMENUACTIVEXCONTROL1.nativeObject
oRadialMenu.BeginUpdate()
oRadialMenu.Template = [SelBackAlpha(1) = 32] // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = [SelBackAlpha(2) = 128] // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.Template = [SelForeColor(3) = 0] // oRadialMenu.SelForeColor(3) = 0x0
oRadialMenu.Template = [RadialLineSize(8) = -1] // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = [RadialLineAlpha(8) = 32] //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = [RadialLineColor(11) = -1] //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.Template = [SelectedIndex(3) = 0] // oRadialMenu.SelectedIndex(3) = 0
oRadialMenu.EndUpdate()
```

XBasic (Alpha Five)

```
' Notifies once the user selects an item.
function SelectItem as v (Item  as  OLE::Exontrol.RadialMenu.1::IItem)
   ' SelectedIndex(3) = Item.Index
   oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
end function
```

```
Dim oRadialMenu as P

oRadialMenu = topparent:CONTROL_ACTIVEX1.activex
oRadialMenu.BeginUpdate()
oRadialMenu.Template = "SelBackAlpha(1) = 32" // oRadialMenu.SelBackAlpha(1) =
32
oRadialMenu.Template = "SelBackAlpha(2) = 128" // oRadialMenu.SelBackAlpha(2) =
128
oRadialMenu.Template = "SelForeColor(3) = 0" // oRadialMenu.SelForeColor(3) = 0
oRadialMenu.Template = "RadialLineSize(8) = -1" // oRadialMenu.RadialLineSize(8) =
-1
oRadialMenu.Template = "RadialLineAlpha(8) = 32" //
oRadialMenu.RadialLineAlpha(8) = 32
oRadialMenu.Template = "RadialLineColor(11) = -1" //
oRadialMenu.RadialLineColor(11) = -1
oRadialMenu.Expanded = .t.
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item
8"
oRadialMenu.Template = "SelectedIndex(3) = 0" // oRadialMenu.SelectedIndex(3) = 0
oRadialMenu.EndUpdate()
```

Visual Objects

```
METHOD OCX_Exontrol1SelectItem(Item) CLASS MainDialog
   // SelectItem event - Notifies once the user selects an item.
   // SelectedIndex(3) = Item.Index

RETURN NIL



oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:[SelBackAlpha,exRadialItems] := 32
oDCOCX_Exontrol1:[SelBackAlpha,exRadialSubItems] := 128
oDCOCX_Exontrol1:[SelForeColor,exRadialFullItems] := RGB(0,0,0)
oDCOCX_Exontrol1:[RadialLineSize,exRadialHotParent] := -1
```

```
oDCOCX_Exontrol1:[RadialLineAlpha,exRadialHotParent] := 32
oDCOCX_Exontrol1:[RadialLineColor,exRadialHotFullItem] := -1
oDCOCX_Exontrol1:Expanded := true
oDCOCX_Exontrol1:Items:ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
oDCOCX_Exontrol1:[SelectedIndex,exRadialFullItems] := 0
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
/*begin event SelectItem(oleobject  Item) - Notifies once the user selects an item.*/
/*
    SelectedIndex(3) = Item.Index
    oRadialMenu = ole_1.Object
*/
/*end event SelectItem*/

OleObject oRadialMenu

oRadialMenu = ole_1.Object
oRadialMenu.BeginUpdate()
oRadialMenu.SelBackAlpha(1,32)
oRadialMenu.SelBackAlpha(2,128)
oRadialMenu.SelForeColor(3,RGB(0,0,0))
oRadialMenu.RadialLineSize(8,-1)
oRadialMenu.RadialLineAlpha(8,32)
oRadialMenu.RadialLineColor(11,-1)
oRadialMenu.Expanded = true
oRadialMenu.Items.ToString = "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item 7,Item 8"
oRadialMenu.SelectedIndex(3,0)
oRadialMenu.EndUpdate()
```

Visual DataFlex

```
// Notifies once the user selects an item.
```

```
Procedure OnComSelectItem Variant   llItem
   Forward Send OnComSelectItem llItem
   // SelectedIndex(3) = Item.Index
End_Procedure

Procedure OnCreate
   Forward Send OnCreate
   Send ComBeginUpdate
   Set ComSelBackAlpha OLEexRadialItems to 32
   Set ComSelBackAlpha OLEexRadialSubItems to 128
   Set ComSelForeColor OLEexRadialFullItems to (RGB(0,0,0))
   Set ComRadialLineSize OLEexRadialHotParent to -1
   Set ComRadialLineAlpha OLEexRadialHotParent to 32
   Set ComRadialLineColor OLEexRadialHotFullItem to -1
   Set ComExpanded to True
   Variant voItems
   Get ComItems to voItems
   Handle hoItems
   Get Create (RefClass(cComItems)) to hoItems
   Set pvComObject of hoItems to voItems
      Set ComToString of hoItems to "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
   Send Destroy to hoItems
   Set ComSelectedIndex OLEexRadialFullItems to 0
   Send ComEndUpdate
End_Procedure
```

XBase++

```
PROCEDURE OnSelectItem(oRadialMenu,Item)
   /*SelectedIndex(3) = Item.Index*/

RETURN

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

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

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

   oRadialMenu := XbpActiveXControl():new( oForm:drawingArea )
   oRadialMenu:CLSID  := "Exontrol.RadialMenu.1" /*{1604BDE1-D48F-4D3F-B51B-
49C0CD74236C}*/
   oRadialMenu:create(,, {10,60},{610,370} )

      oRadialMenu:SelectItem := {|Item| OnSelectItem(oRadialMenu,Item)} /*Notifies
once the user selects an item.*/

      oRadialMenu:BeginUpdate()
      oRadialMenu:SetProperty("SelBackAlpha",1/*exRadialItems*/,32)
      oRadialMenu:SetProperty("SelBackAlpha",2/*exRadialSubItems*/,128)

oRadialMenu:SetProperty("SelForeColor",3/*exRadialFullItems*/,AutomationTranslateC
 GraMakeRGBColor  ( { 0,0,0 } )  , .F. ))
      oRadialMenu:SetProperty("RadialLineSize",8/*exRadialHotParent*/,-1)
      oRadialMenu:SetProperty("RadialLineAlpha",8/*exRadialHotParent*/,32)
      oRadialMenu:SetProperty("RadialLineColor",11/*exRadialHotFullItem*/,-1)
      oRadialMenu:Expanded := .T.
      oRadialMenu:Items():ToString := "Item 1,Item 2,Item 3,Item 4,Item 5,Item 6,Item
7,Item 8"
      oRadialMenu:SetProperty("SelectedIndex",3/*exRadialFullItems*/,0)
      oRadialMenu:EndUpdate()

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

```
   ENDDO
RETURN
```

event SelectParent ()

Occurs once the user clicks the parent of the item.

Type	Description

The SelectParent event occurs once the user clicks the parent of the item. The SelectItem event notifies once the user selects an item. The BrowseItem event notifies when a new item has been selected / browsed. When the user selects a new item, it is displayed on the parent portion of the control, while its content / children is displayed around. The AllowToggleExpand property specifies whether the radial menu can be shown in collapsed state. The Parent item property specifies the parent item.

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

C#

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

VB

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

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

C#

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

C++

```
void OnSelectParent()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event SelectParent()

end event SelectParent
```

VB.NET

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

VB6

```
Private Sub SelectParent()
End Sub
```

VBA

```
Private Sub SelectParent()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnSelectParent(oRadialMenu)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComSelectParent
    Forward Send OnComSelectParent
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_SelectParent()
{
}
```

XBasic

```
function SelectParent as v ()
end function
```

dBASE

```
function nativeObject_SelectParent()
return
```