ExCalendar

The ExCalendar Library contains two components, simple version and drop-down version that allow you to select a date with a nice GUI. You can select the date between 1/1/100 and 12/31/9999. Both versions support images, colors, font attributes, tooltips for any date. The ExCalendar 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.

Features include:

- Skinnable Interface Support (ability to apply a skin to any background part)
- Unlimited options to show any colors, icons, captions on any date
- Easy way to define the control's visual appearance in design mode, using XP-Theme elements or EBN objects
- ISO8601 compatible
- Repetitive-events support (RFC 5545)
- Print and Print Preview support
- DATE and TIME editing support
- Simple and dropdown versions
- Single or multiple months in the client area
- Single or multiple selection
- Ability to limit the date being displayed and selected
- Easy way to scroll to next/previous month, change year, and select date via a simple click
- Date spinner support, for the dropdown version
- Color, font attributes, images, markers, for any date
- Multi-lines HTML tooltip support
- Easy way to load/save the event date collection
- Customize data format for drop-down version
- Ability to load icons from BASE64 encoded strings
- Supports BMP, EMF, EXIF, GIF, ICON, JPEG, PNG, TIFF or WMF formats on dates or on the control's background
- Flat or 3D appearance
- Supports multiple languages
- Auto size or fixed size
- 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 AlignmentEnum

The AlignmentEnum type specifies the alignment of the day.

Name	Value	Description
LeftAlignment	0	The source is left aligned.
CenterAlignment	1	The source is centered.
RightAlignment	2	The source is centered.

constants AppearanceDayEnum

Defines how the date is displayed.

Name	Value	Description
DayFlat	0	Flat
Day3D	1	3D

constants AppearanceEnum

Defines the way how the control's border looks like.

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 AutoAdvanceEnum

The AutoAdvanceEnum expression specifies if the focus element is moving to the next element. Use the AutoAdvance property to disable moving the focused field, while user type data.

Name	Value	Description
exAdvanceNone	0	The focused element is not moved to the next position.
exAdvanceCycle	1	Moves the focus to the next element. If the focused element is the last displayed element, the fist element is displayed.
exAdvanceLast	2	Moves the focus to the next element until the focused element is the last displayed element

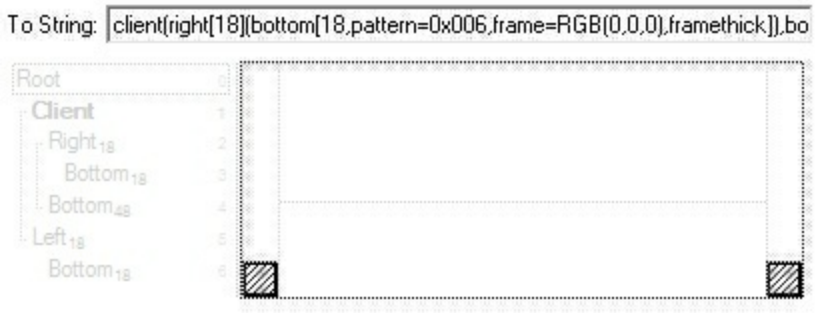
constants AutoSizeEnum

The AutoSizeEnum type specifies the way the control arranges the days within the calendar's view area. Use the AutoSize property to specify the way the control arranges the days in the control.

Name	Value	Description
exFontSize	-1	(default) The size of the control's font specifies the size for the days in the calendar. The Font property specifies the control's font. The MaxMonthX property specifies the number of months that can displayed on the horizontal axis. The MaxMonthY property specifies the number of months that can displayed on the vertical axis.
exFixedSize	0	The days in the calendar has a fixed size. The FixedCellWidth property specifies the width to display a day in the calendar control. The FixedCellHeight property specifies the height to display a day in the calendar control. The MaxMonthX property specifies the number of months that can displayed on the horizontal axis. The MaxMonthY property specifies the number of months that can displayed on the vertical axis. For instance, if the AutoSize property is exFixedSize, and the MaxMonthX is 1, and the MaxMonthY is 1, the calendar is always displaying 1 month in the client's area.
exFitClient	1	The months on horizontal and vertical axis should fit the control's client area. The size required to display a day in the calendar is computed so all months fit the control's client area. The MaxMonthX property specifies the number of months that are displayed on the horizontal axis. The MaxMonthY property specifies the number of months that are displayed on the vertical axis. For instance, if the AutoSize property is exFitClient, and the MaxMonthX is 2, and the MaxMonthY is 2, the calendar is always displaying 4 months in the client's area.

constants BackgroundExtPropertyEnum

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

Name	Value	Description
exToStringExt	0	Specifies the part's ToString representation. The BackgroundExt property specifies the EBN String format to be displayed on the object's background. *The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field.* *Sample:* *"client(right[18](bottom[18,pattern=6,frame=0,framethick]),bottom[4* *(bottom[18,pattern=6,frame=0,framethick])"* *generates the following layout:*  *where it is applied to an object it looks as follows:*  *(String expression, read-only).*

exBackColorExt	1	Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000.* *(Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*
		Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect.* *Based on the exAnchorExt value the exClientExt is:* • *0 (**none**, the object is not anchored to any side), the format of the exClientExt is **"left,top,width,height"** (as string) where (left,top) margin indicates the position where the part starts, and the (width,height) pair specifies its size. The left, top, width or height could be any expression (+,-,/ or *) that can include numbers associated with pixels or percents. For instance: "25%,25%,50%,50%" indicates the middle of the parent object, and so when the parent is resized the client is resized accordingly. The "50%-8,50%-8,16,16" value specifies that the size of the object is always 16x16 pixels and positioned on the center of the parent object.* • *1 (**left**, the object is anchored to left side of the parent), the format of the exClientExt is **width** (string or numeric) where width indicates the width of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates*

exClientExt	2	*the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *2 (**right**, the object is anchored to right side of the parent object), the format of the exClientExt is **width** (string or numeric) where width indicates the width of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *3 (**client**, the object takes the full available area of the parent), the exClientExt has no effect.* • *4 (**top**, the object is anchored to the top side of the parent object), the format of the exClientExt is **height** (string or numeric) where height indicates the height of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *5 (**bottom**, the object is anchored to bottom side of the parent object), the format of the exClientExt is **height** (string or numeric) where height indicates the height of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* *Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent.*

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

FormatAnchor property customizes the visual effect for anchor elements.

- ** ... ** 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.
exTextExt	4	• **<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

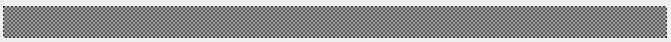
a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

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

exTextExtAlignment	6	*vertically aligned in the middle, and horizontally aligned on the left.* • *17, (hexa 0x11, **Middle-Center**), Text is vertically aligned in the middle, and horizontally aligned at the center.* • *18, (hexa 0x12, **Middle-Right**), Text is vertically aligned in the middle, and horizontally aligned on the right.* • *32, (hexa 0x20, **Bottom-Left**), Text is vertically aligned at the bottom, and horizontally aligned on the left.* • *33, (hexa 0x21, **Bottom-Center**), Text is vertically aligned at the bottom, and horizontally aligned at the center.* • *34, (hexa 0x22, **Bottom-Right**), Text is vertically aligned at the bottom, and horizontally aligned on the right.* *(Numeric expression)*
		Indicates the pattern to be shown on the object. The exPatternColorExt specifies the color to show the pattern. *The valid values for exPatternExt are:* • *0, (hexa 0x000, **Empty**), The pattern is not visible* • *1, (hexa 0x001, **Solid**),*  • *2, (hexa 0x002, **Dot**),*  • *3, (hexa 0x003, **Shadow**),*  • *4, (hexa 0x004, **NDot**),*  • *5, (hexa 0x005, **FDiagonal**),*  • *6, (hexa 0x006, **BDiagonal**),*  • *7, (hexa 0x007, **DiagCross**),* 

exPatternExt	7	• *8, (hexa 0x008, **Vertical**),* • *9, (hexa 0x009, **Horizontal**),*  • *10, (hexa 0x00A, **Cross**),*  • *11, (hexa 0x00B, **Brick**),*  • *12, (hexa 0x00C, **Yard**),*  • *256, (hexa 0x100, **Frame**),*  *. The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags.* • *768, (hexa 0x300, **FrameThick**),*  *. The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags.* *(Numeric expression)*
exPatternColorExt	8	Indicates the color to show the pattern on the object. The exPatternColorExt property has effect only if the exPatternExt property is not 0 (empty). The exFrameColorExt specifies the color to show the frame (the exPatternExt property includes the exFrame or exFrameThick flag) *(Color expression)*
exFrameColorExt	9	Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. *(Color expression)*
exFrameThickExt	10	Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property.

		(Boolean expression)
exUserDataExt	11	Specifies an extra-data associated with the object. (*Variant expression)*

constants BackgroundPartEnum

The BackgroundPartEnum type indicates parts in the control. Use the Background, 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
exScrollUp	0	Specifies the visual appearance for the up arrow in the calendar's header.
exScrollDown	1	Specifies the visual appearance for the down arrow in the calendar's header.
exScrollLeft	2	Specifies the visual appearance for the left arrow in the calendar's header.
exScrollRight	3	Specifies the visual appearance for the right arrow in the calendar's header.
exDropDownButtonUp	4	Returns or sets a value that indicates the visual drop down button.
exDropDownButtonDown	5	Returns or sets a value that indicates the visual drop down button.
exDaysHeader	6	Specifies the visual appearance for the days header.
exWeeksHeader	7	Specifies the visual appearance for the weeks header.
exDateHeader	8	Specifies the visual appearance for the months header in a calendar control.
exDateTodayUp	9	Specifies the visual appearance for the today button in a calendar control, when it is up.
exDateTodayDown	10	Specifies the visual appearance for the today button in a calendar control, when it is down.
exDateScrollThumb	11	Specifies the visual appearance for the scrolling thumb in a calendar control.
exDateScrollRange	12	Specifies the visual appearance for the scrolling range in a calendar control.
exDateSeparatorBar	13	Specifies the visual appearance for the separator

		bar in a calendar control.
exMarkToday	14	Returns or sets a value that indicates the visual appearance for today date.
exMonthSelect	15	Specifies the visual appearance for the selected month in the months drop down window.
exMonthSelectForeColor	16	Specifies the foreground color for the selected month in the months drop down window.
exDropDownBackColor	17	Specifies the background color for the drop down portion of the control. This option does not support EBN objects.
exDropDownForeColor	18	Specifies the foreground color for the drop down portion of the control.
exDropDownSelBackColor	19	Specifies the background color for the selected date in the drop down portion of the control.
exDropDownSelForeColor	20	Specifies the foreground color for the selected date in the drop down portion of the control.
exDateHeaderForeColor	21	Specifies the foreground color to show the months in the header.
exSpinUpButtonUp	22	Specifies the visual appearance for the up spin button when it is not pressed.
exSpinUpButtonDown	23	Specifies the visual appearance for the up spin button when it is pressed.
exSpinDownButtonUp	24	Specifies the visual appearance for the down spin button when it is not pressed.
exSpinDownButtonDown	25	Specifies the visual appearance for the down spin button when it is pressed.
exDaysHeaderForeColor	28	Specifies the foreground color for the days header.
exWeeksHeaderForeColor	29	Specifies the foreground color for the weeks header.
exMarkTodayForeColor	30	Specifies the foreground color for the today date.
exDateTodayForeColor	31	Specifies the foreground color for the Today button.
exFocusDate	32	Specifies visual appearance for the focused date.
exFocusDateForeColor	33	Specifies foreground color for the focused date.
		Indicates the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip

exToolTipAppearance	64	remains visible if the mouse pointer is stationary within a control. 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.
exDropDownAppearance	67	Specifies the visual appearance of the drop down portion of the control. This option can be a value of AppearanceEnum type, 255, or an EBN identifier. If exDropDownAppearance is 255, the drop down portion of the control shows a shadow frame.
exSelBackColorUnFocus	68	Specifies the background color for selected object when the control loses the focus.
exSelForeColorUnFocus	69	Specifies the foreground color for selected object when the control loses the focus.
exMarkerColor	187	Specifies the color or the visual appearance to apply on dates with Marker property set.

constants CheckStateEnum

Specifies the states of the control's check box. Use the AllowCheckBox property to display a checkbox on the control.

Name	Value	Description
exUnchecked	0	Specifies whether the date is unchecked.
exChecked	1	Specifies whether the date is checked.

constants FormatDateEnum

Defines the format of the displayed date into CalendarCombo control.

Name	Value	Description
ShortDate	0	ShortDate. The date is displayed like: "12/31/1971"
LongDate	1	LongDate. The date is displayed like: "January 31 1971, Friday"
UserDate	2	UserDate. The user defines how the date is displayed

constants HeaderArrowEnum

The HeaderArrowEnum expression specifies the arrow being clicked in the header of the calendar. The DataChanging event notifies your application that the user is about to change the browsed date.

Name	Value	Description
exPrevYear	1	The previous year arrow is clicked.
exNextYear	2	The next year arrow is clicked.
exPrevMonth	4	The previous month arrow is clicked.
exNextMonth	8	The next month arrow is clicked.

constants HeaderFieldEnum

The HeaderFieldEnum type defines the predefined-fields to be displayed by the DateHeaderFormat property. The DateHeaderFormat property specifies the CRD format to display the month/year/buttons within the date's header. The DateHeaderField property specifies the HTML caption to be shown on the giving field of the date's header. The HeaderFieldEnum type supports the following values:

Name	Value	Description
exHeaderDate	1	Displays the month/year of the browsed date. Clicking the part shows the control's months selector, so the user can select a different month/year.
exHeaderPrevMonth	2	Displays the "-" character. Clicking the part navigates the control to previously month.
exHeaderNextMonth	3	Displays the "+" character. Clicking the part navigates the control to the next month.
exHeaderPrevYear	4	Displays the "<" character. Clicking the part navigates the control to previously year.
exHeaderNextYear	5	Displays the ">" character. Clicking the part navigates the control to the next year.

constants LineStyleEnum

Defines the types of line used by control.

Name	Value	Description
NoLine	-1	No line
SmallDots	1	SmallDots
LargeDots	2	LargeDots
Solid	0	Solid

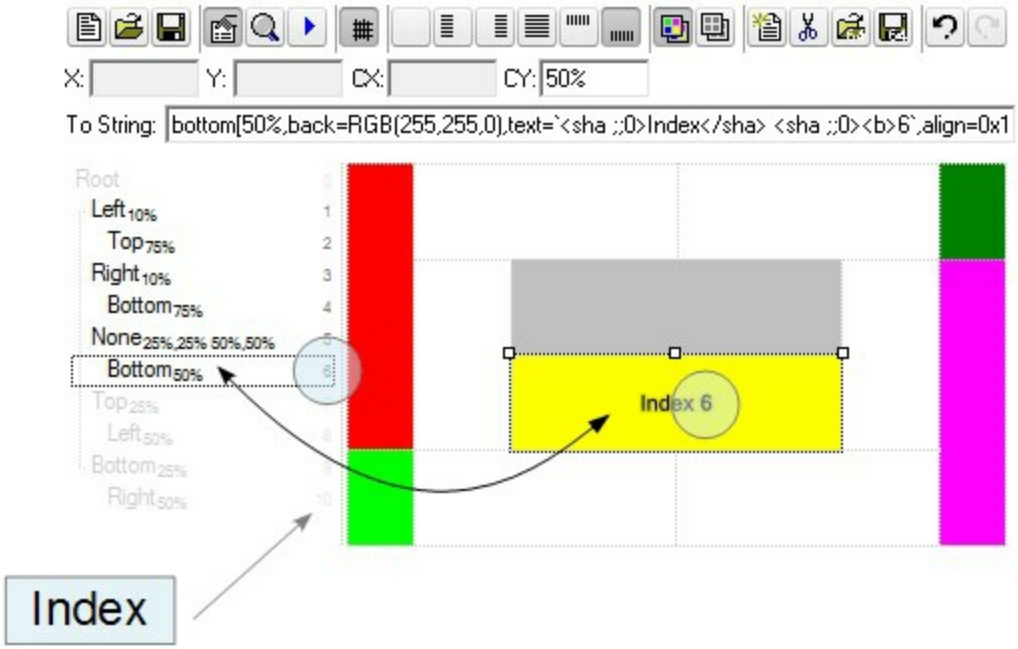
constants MonthEnum

Defines the months.

Name	Value	Description
January	1	January
February	2	February
March	3	March
April	4	April
May	5	May
June	6	June
July	7	July
August	8	August
September	9	September
October	10	October
November	11	November
December	12	December

constants IndexExtEnum

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

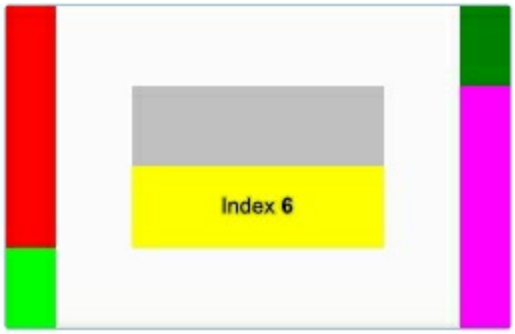


In this sample, there are 11 objects that composes the EBN, so the Index property goes from 0 which indicates the root, and 10, which is the last item in the list

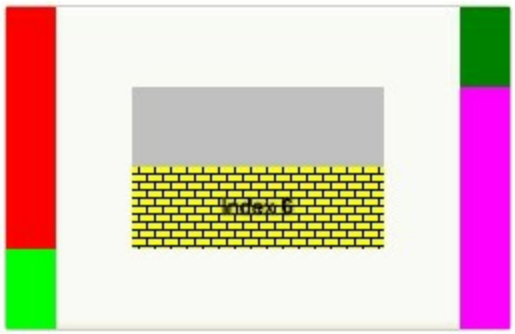
So, let's apply this format to an object, to change the exPatternExt property for the object with the Index 6:

Before calling the BackgroundExt property:

After calling the BackgroundExt property:



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

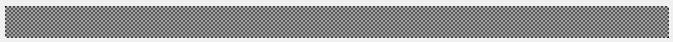
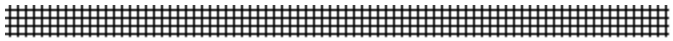
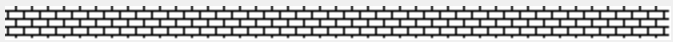


The IndexExtEnum type supports the following values:

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

constants PatternEnum

The PatternEnum expression indicates the type of brush. Use the NonworkingDaysPattern property to specify the pattern to fill non-working days.

Name	Value	Description
exPatternEmpty	0	The pattern is not visible.
exPatternSolid	1	
exPatternDot	2	
exPatternShadow	3	
exPatternNDot	4	
exPatternFDiagonal	5	
exPatternBDiagonal	6	
exPatternDiagCross	7	
exPatternVertical	8	
exPatternHorizontal	9	
exPatternCross	10	
exPatternBrick	11	
exPatternYard	12	

constants PictureDisplayEnum

Specifies how a picture object is displayed.

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

constants UIChangeEnum

The UIChangeEnum type specifies the states and parts of the control's UI (user interface) that are signaled through the UIChange event. The UIChangeEnum type supports the following predefined values:

Name	Value	Description
exCheckDate	0	The date has been checked.
exUncheckDate	1	The date has been un-checked.
exSpinUpDate	2	The spin-up button has been pressed.
exSpinDownDate	3	The spin-down button has been pressed.
exClickDropDownDate	4	The drop-down button has been clicked.
exSelectDate	5	The user selects a new date from the drop down calendar.
exChangeDate	6	A new date is being displayed in the drop down calendar window.
exLeftDropDownDate	17	The user clicks the left arrow button on the drop down portion of the control.
exRightDropDownDate	18	The user clicks the right arrow button on the drop down portion of the control.
exUpDropDownDate	20	The user clicks the up arrow button on the drop down portion of the control.
exDownDropDownDate	24	The user clicks the down arrow button on the drop down portion of the control.

constants UIVisualThemeEnum

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

Name	Value	Description
exNoVisualTheme	0	exNoVisualTheme
exDefaultVisualTheme	16777215	exDefaultVisualTheme
exButtonsVisualTheme	4	exButtonsVisualTheme
exCalendarVisualTheme	8	exCalendarVisualTheme
exSpinVisualTheme	32	exSpinVisualTheme
exCheckBoxVisualTheme	64	exCheckBoxVisualTheme

constants VisibleEnum

The VisibleEnum type specifies whether the component displays the object. The AllowSpin, AllowCheckBox property specifies whether the control's label display the spin or a check-box. The VisibleEnum type supports the following values:

Name	Value	Description
exHidden	0	The object is hidden.
exVisible	-1	The object is always visible.
exVisibleOnFocus	1	The object is visible while the control gets the focus.

constants WeekDayEnum

Defines the week days.

Name	Value	Description
Sunday	1	Sunday
Monday	2	Monday
Tuesday	3	Tuesday
Wednesday	4	Wednesday
Thursday	5	Thursday
Friday	6	Friday
Saturday	7	Saturday

constants WeekNumberAsEnum

The WeekNumberAsEnum type specifies the ways the control displays the week number for dates. The ShowWeeks property specifies whether the week number header is shown or hidden. The DisplayWeekNumberAs property specifies the way the control displays the week number. The FirstDay property specifies the first day of the week where the week begins. The WeekNumberAsEnum type supports the following values:

Name	Value	Description
exISO8601WeekNumber	0	Indicates that the week number is displayed according to the ISO8601 standard, which specifies that the first week of the year is the one that includes the January the 4th (default)
exSimpleWeekNumber	1	The first week starts on January 1st of a given year, week n+1 starts 7 days after week n

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.

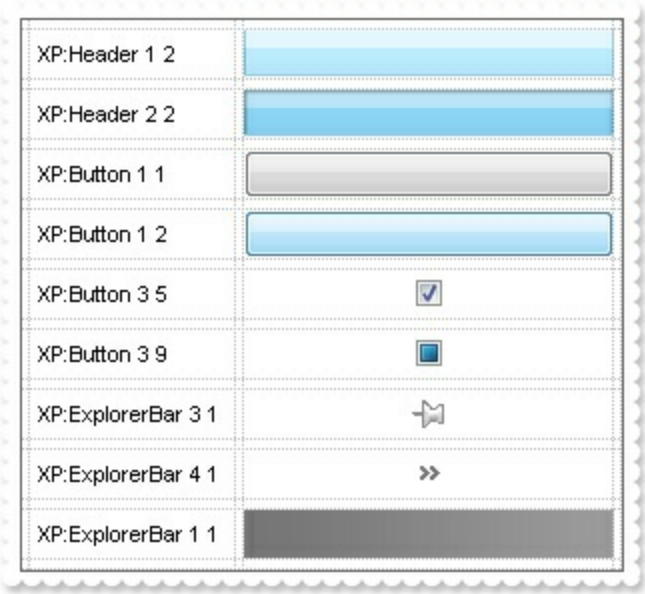
method Appearance.Add (ID as Long, Skin as Variant)

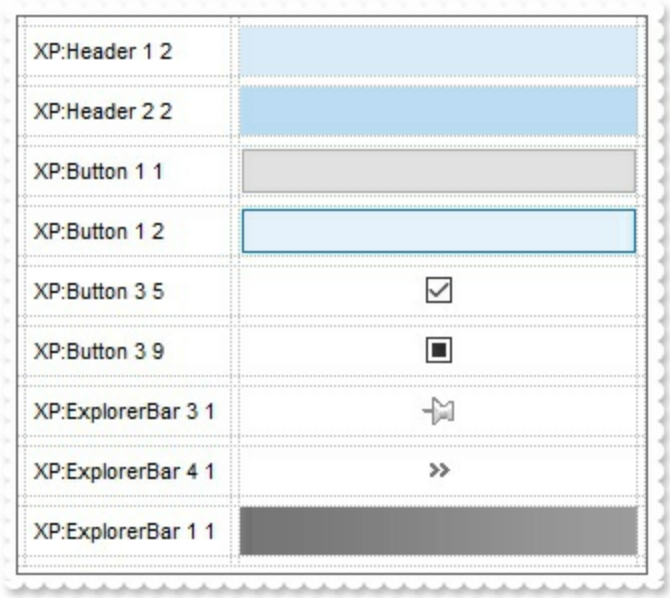
Adds or replaces a skin object to the control.

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:



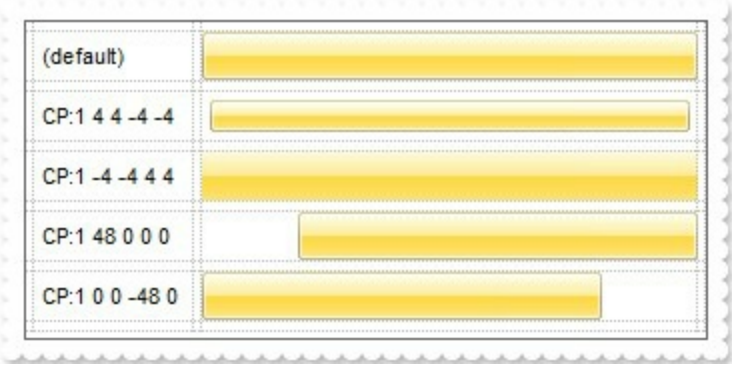


Skin as Variant

- 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.

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.

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

- borders, Appearance property
- months, weeks, days header, Background property, HeaderBackColor property,
- up down, left or right arrows, Background property
- selected date(s), SelBackColor property
- events, BackColor property
- Today button, scrolling dates area, Background property
- today date, MarkToday property
- selected items in the months selector, Background property
- drop down button, scrollbars, tooltips, Background property

For instance, the following VB sample changes the visual appearance for the selected date. The SelBackColor property indicates the selection background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the SelBackColor property to indicates the index of the skin that we want to use. The

sample applies the "[ ]" to the selected date(s):


```
With Calendar1
   With .VisualAppearance
      .Add &H23, "D:\Temp\ExCalendar.Help\seldate.ebn"
   End With
   .SelBackColor = &H23000000
End With
```

The following C++ sample changes the visual appearance for selected date:

```
m_calendar.GetVisualAppearance().Add( 0x23,
COleVariant("D:\\Temp\\ExCalendar.Help\\seldate.ebn"));
m_calendar.SetSelBackColor( 0x23000000 );
```

The following VB.NET sample changes the visual appearance for selected date:

```
With AxCalendar1
   With .VisualAppearance
      .Add(&H23, "D:\Temp\ExCalendar.Help\seldate.ebn")
   End With
   .Template = "SelBackColor = 587202560"
End With
```

where the 587202560 value in hexa representation is &H23000000

The following C# sample changes the visual appearance for selected date:

```
axCalendar1.VisualAppearance.Add(0x23, "D:\\Temp\\ExCalendar.Help\\seldate.ebn");
axCalendar1.Template = "SelBackColor = 587202560";
```

where the 587202560 value in hexa representation is 0x23000000

The following VFP sample changes the visual appearance for selected date:

```
With thisform.Calendar1
   With .VisualAppearance
      .Add(35, "D:\Temp\ExCalendar.Help\seldate.ebn")
   EndWith
   .SelBackColor = 587202560
EndWith
```

where the 587202560 value in hexa representation is 0x23000000, 35 is 0x23 in hexa

The screen shot was generated using the following template:

```
VisualAppearance
{
   Add( 1,
"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK
 )
   Add( 2,
"gBFLBCJwBAEHhEJAEGg4Ba4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK
 )

   Add(3,
"gBFLBCJwBAEHhEJAEGg4BW4Cg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK
 )

   Add( 4,
"gBFLBCJwBAEHhEJAEGg4BDgGg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUAIIRZG

   Add( 5,
"gBFLBCJwBAEHhEJAEGg4BD4Gg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUAIIRZG
 )

Add(6,"gBFLBCJwBAEHhEJAEGg4BWMIQAAYAQGKIYBkAKBQAGaAoDDMOILQiMQxDPBMK



Add(7,"gBFLBCJwBAEHhEJAEGg4BW8IQAAYAQGKIYBkAKBQAGaAoDDMOILQiMQxDPBMK
```

```
    Add(8,
"gBFLBCJwBAEHhEJAEGg4BDAGg6AADACAxRDAMgBQKAAzAFBIYhxASCBhGaCYUACCIVRI



Add(9,"gBFLBCJwBAEHhEJAEGg4BIAGg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA



}

AutoSize = False
DrawGridLine = 1
ShowWeeks = True
HeaderForeColor = 0
SelBackColor = 33554432
Background(0) = 100663296
Background(1) = 117440512
Background(2) = 67108864
Background(3) = 83886080
Background(4) = 134217728
Background(5) = 150994944
Background(6) = 50331648
Background(7) = 50331648
Background(8) = 16777216
Background(9) = 16777216
Background(10) = 33554432
Background(11) = 33554432
Background(12) = 33554432
Background(13) = 16777216
Background(15) = 16777216
```

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

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

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

- borders, Appearance property
- months, weeks, days header, Background property, HeaderBackColor property,
- up down, left or right arrows, Background property
- selected date(s), SelBackColor property
- events, BackColor property
- Today button, scrolling dates area, Background property
- today date, MarkToday property
- selected items in the months selector, Background property
- drop down button, scrollbars, tooltips, Background property

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

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

- borders, Appearance property
- months, weeks, days header, Background property, HeaderBackColor property,
- up down, left or right arrows, Background property
- selected date(s), SelBackColor property
- events, BackColor property
- Today button, scrolling dates area, Background property
- today date, MarkToday property
- selected items in the months selector, Background property
- drop down button, scrollbars, tooltips, Background property

property Appearance.RenderType as Long

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

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

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

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

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

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

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

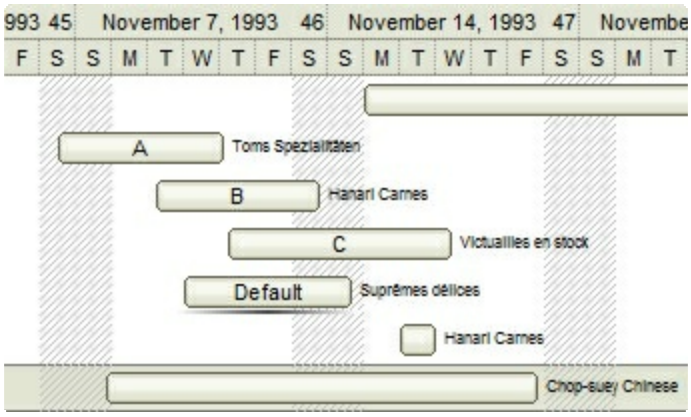
In the following screen shot the following objects displays the current EBN with a different color:

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

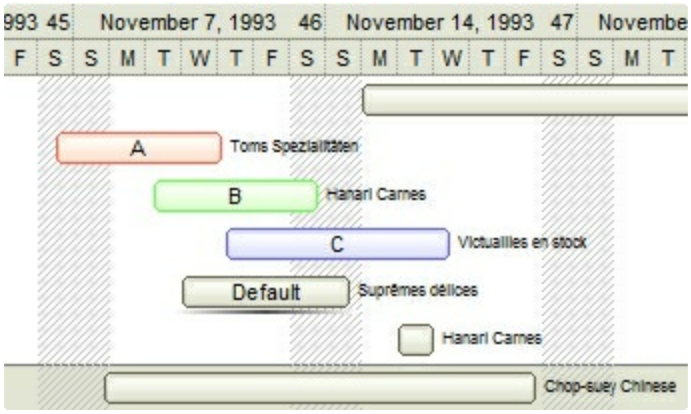
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

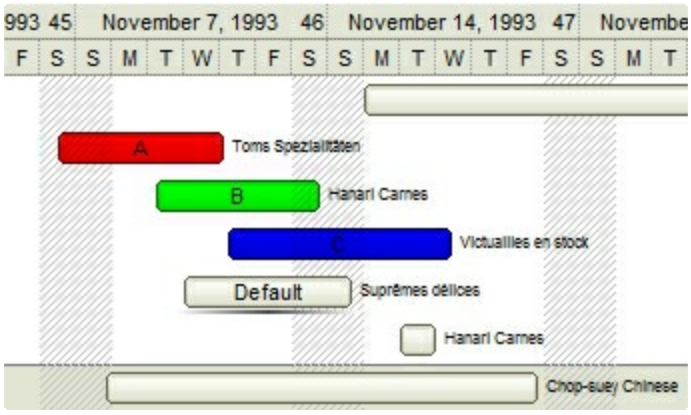
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



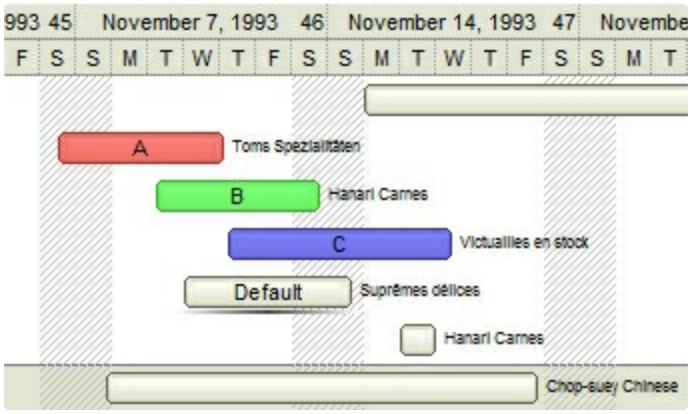
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the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

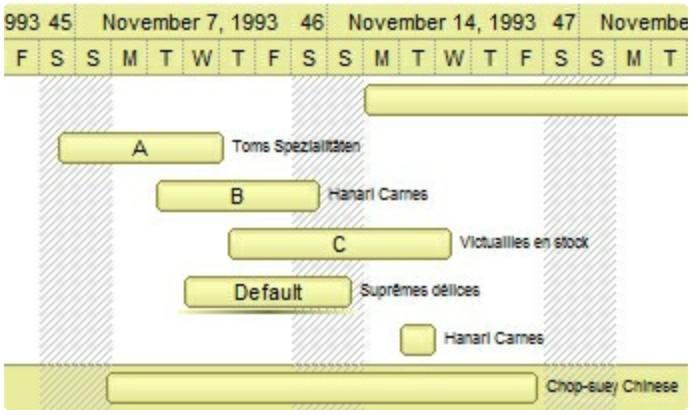


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

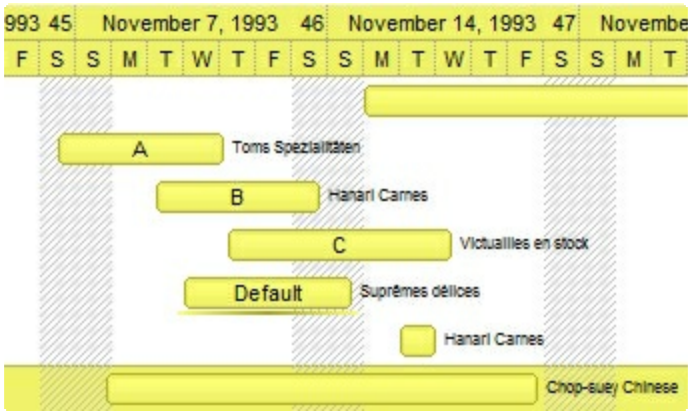


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

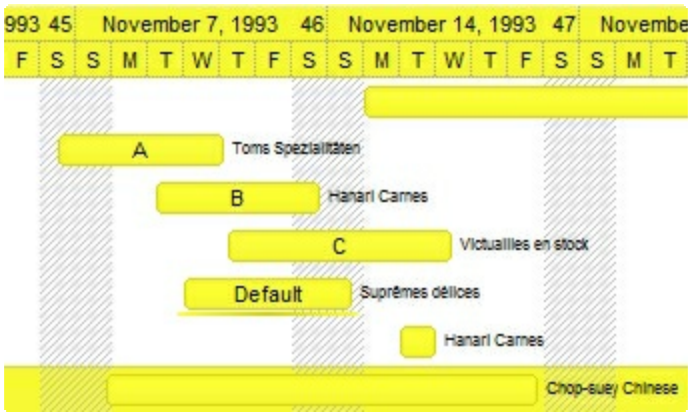
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



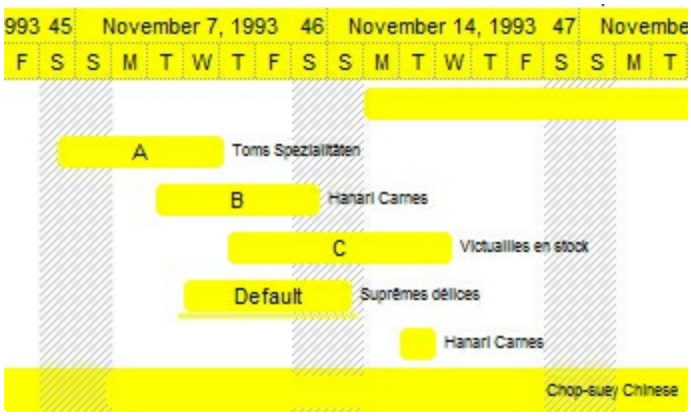
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



*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):*


993 45
November 7, 1993
46
November 14, 1993
47
Novembe
F S S M T W T F S S M T W T F S S M T
A
Toms Spezialitäten
B
Hanari Carnes
C
Victuailles en stock
Default
Suprêmes délices
Hanari Carnes
Chop-suey Chinese

Calendar 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: {D8F4D09C-3FD1-4479-ABA3-4F195C20050C}. The object's program identifier is: "Exontrol.Calendar". The /COM object module is: "ExCalendar.dll"

The ExCalendar Library contains two ActiveX controls, simple version and drop-down version that allow you to select a date with a nice GUI. You can select the date between 1/1/100 and 12/31/9999. Both versions support images, colors, font attributes, tooltips for any date.

Features include:

- simple and dropdown versions.
- skinnable interface
- multiple months in the client area.
- single or multiple selection
- easily scroll to next/previous month, change year, and select date via a simple click
- color, font attributes, tooltips, images, markers, for any date
- easy way to load/save the event date collection
- customize data format for drop-down version
- flat or 3D appearance.
- supports multiple languages
- auto size or fixed size.

The Calendar object supports the following properties and methods:

Name	Description
AlignmentDay	Specifies the alignment of the days within the control.
Appearance	Retrieves or sets the control's appearance
AppearanceDay	Retrieves or sets a value that determines the day's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoSize	Retrieves or sets a value that indicates whether the control automatically resizes the cell based on the size of the font.
BackColor	Retrieves or sets the control's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance while multiple changes are done at once. This method prevents the control from painting until

	the EndUpdate method is called.
BorderLineColor	Retrieves or sets a value that indicates the border line color.
CommentBackColor	Retrieves or sets the color to highlight the commented events.
Date	Retrieves or sets the browsed date. Ensures that the date is visible.
DateFromPoint	Retrieves the date from point.
DateHeaderField	Specifies the HTML caption to be shown on the giving field of the date's header.
DateHeaderFormat	Specifies the CRD format to display the month/year/buttons within the date's header.
DisplayWeekNumberAs	Specifies the way the control displays the week number.
DoDate	Composes a DATE type, based on year, month and day.
DrawBorderLine	Retrieves or sets a value that indicates the border line style.
DrawGridLine	Retrieves or sets a value that identifies the type of grid lines.
Enabled	Retrieves or sets a value that indicates whether the control is enabled ot disabled.
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.
Events	Retrieves the control date events collection.
ExecuteTemplate	Executes a template and returns the result.
FirstDay	Retrieves or sets a value that indicates the first day of the week.
FirstVisibleDate	Retrieves the first visible date.
FixedCellHeight	Retrieves or sets a value that indicates the cell's height while the AutoSize is exFixed.
FixedCellWidth	Retrieves or sets a value that indicates the cell's width while the AutoSize is exFixed.
FocusDate	Retrieves or sets the focused date
Font	Retrieves or sets the control's font.

ForeColor	Retrieves or sets the control's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FreezeEvents	Prevents the control to fire any event.
GridLineColor	Retrieves or sets a value that indicates the grid lines color.
HeaderBackColor	Retrieves or sets a value that indicates the background color used for weeks and week days headers.
HeaderForeColor	Retrieves or sets a value that indicates the foreground color used for weeks and week days headers.
HideSelection	Specifies whether selected date appears selected when a control loses focus.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets the control's handle image list.
ImageSize	Retrieves or sets the size of icons the control displays..
IntegralHeight	Retrieves the height of the control to fit the MaxMonthY months in the client area.
IntegralWidth	Retrieves the width of the control to fit the MaxMonthX months in the client area.
LastVisibleDate	Retrieves the last visible date.
LocAMPM	Retrieves the time marker such as AM or PM using the current user regional and language settings.
LocFirstDay	Indicates the first day of the week, as specified in the regional settings.
Locked	Specifies whether the user can change the selection.
LocMonthNames	Retrieves the list of month names, as indicated in the regional settings, separated by space.
LocWeekDays	Retrieves the list of names for each week day, as indicated in the regional settings, separated by space.
MarkToday	Retrieves or sets a value that indicates whether the control marks the today date.
MaxDate	Retrieves or sets the min date.
MaxMonthX	Specifies the maximum number of months horizontally displayed.
	Specifies the maximum number of months vertically

MaxMonthY	displayed.
MaxScrollYear	Specifies the maximum year when scrolling.
MinDate	Retrieves or sets the min date.
MinMonthX	Specifies the minimum number of months horizontally displayed.
MinMonthY	Specifies the minimum number of months vertically displayed.
MinScrollYear	Specifies the minimum year when scrolling.
MonthName	Retrieves or sets the month's name.
MonthNames	Retrieves or sets a value that indicates the list of month names, separated by space.
NonMonthDaysColor	Retrieves or sets a value that indicates the color to show the non-month days.
NonworkingDays	Retrieves or sets a value that indicates the non-working days, for each week day a bit.
NonworkingDaysColor	Retrieves or sets a value that indicates the color to fill the non-working days.
NonworkingDaysForeColor	Retrieves or sets a value that indicates the foreground color for non-working days.
NonworkingDaysPattern	Retrieves or sets a value that indicates the pattern being used to fill non-working days.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
Refresh	Refreshes the control.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
SelCount	Retrieves the count of selected dates.
SelDate	Selects a date while SingleSel is true.
SelectDate	Retrieves the selected date, given its index into selected dates collection. Use SelCount in order to get the count of selected dates.
Selection	Serializes the selected dates to a string.
SelectTodayDate	Specifies whether the current date is selected when the

	user clicks the Today button.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowDays	Retrieves or sets a value that indicates whether the week days header is visible or hidden.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowMonth	Retrieves or sets a value that indicates whether the month header is visible or hidden.
ShowMonthSelector	Retrieves or sets a value that indicates whether the user is able to select a new month by clicking in the month header.
ShowNonMonthDays	Specifies whether the control displays the dates that are not part of the month.
ShowTodayButton	Retrieves or sets a value that indicates whether the today button is visible or hidden.
ShowToolTip	Shows the specified tooltip at given position.
ShowWeeks	Retrieves or sets a value that indicates whether the weeks header is visible or hidden.
ShowYearScroll	Retrieves or sets a value that indicates whether the scroll bar for changing the year is visible or hidden.
ShowYearSelector	Retrieves or sets a value that indicates whether the year selector is visible or hidden.
SingleSel	Retrieves or sets a value that indicates whether the control supports single or multiple selection.
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.
TodayCaption	Retrieves or sets a value that indicates the today button's caption.
ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipFont	Retrieves or sets the tooltip's font.
	Specifies the period in ms of time the ToolTip remains

ToolTipPopDelay	visible if the mouse pointer is stationary within a control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
UnSelDate	Unselects the date.
UseVisualTheme	Specifies whether the control uses the current visual theme to display certain UI parts.
Value	Specifies the selected date.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.
VisualDesign	Invokes the control's VisualAppearance designer.
WeekDayName	Retrieves or sets a value that indicates the week day short name in the week days header.
WeekDays	Retrieves or sets a value that indicates the list of short names for each week day, separated by space.

property Calendar.AlignmentDay as AlignmentEnum

Specifies the alignment of the days within the control.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the day within the cell.

By default, the AlignmentDay property is RightAlignment, which means that the day is displayed to the right of their cells. Use the AlignmentDay property to specify the horizontal alignment of the day inside the cell where it is displayed. For instance, if the cell's size is fixed using the AutoSize property on exFixedSize, FixedCellWidth and FixedCellHeight property you can specify the day's alignment to be on center.

property Calendar.Appearance as AppearanceEnum

Retrieves or sets the control's appearance

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

Defines how the control's border looks like. Use the Appearance property to remove the control's default border. Use the Add method to add new skins to the control. Use the BackColor property to specify the control's background color. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips.

The following VB sample changes the visual aspect of the borders of the control (please check the above picture for round corners):

```
With Calendar1
  .BeginUpdate
    .VisualAppearance.Add &H16, "c:\temp\frame.ebn"
```

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

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

```
With AxCalendar1
  .BeginUpdate()
  .VisualAppearance.Add(&H16, "c:\temp\frame.ebn")
  .Appearance = &H16000000
  .BackColor = Color.FromArgb(250, 250, 250)
  .EndUpdate()
End With
```

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

```
axCalendar1.BeginUpdate();
axCalendar1.VisualAppearance.Add(0x16, "c:\\temp\\frame.ebn");
axCalendar1.Appearance = (EXCALENDARLib.AppearanceEnum)0x16000000;
axCalendar1.BackColor = Color.FromArgb(250, 250, 250);
axCalendar1.EndUpdate();
```

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

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

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

```
with thisform.Calendar1
  .BeginUpdate
    .VisualAppearance.Add(0x16, "c:\temp\frame.ebn")
    .Appearance = 0x16000000
    .BackColor = RGB(250, 250, 250)
  .EndUpdate
```

endwith

property Calendar.AppearanceDay as AppearanceDayEnum

Retrieves or sets a value that determines the day's appearance.

Type	Description
AppearanceDayEnum	An AppearanceDayEnum expression that defines the date's appearance.

Defines the was how border's date looks like. The control defines two types of date's appearance: flat and 3D.

The following sample displays the control using DayFlat:

The following sample displays the control using Day3D:

method Calendar.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 Calendar1_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 Calendar.AutoSize as AutoSizeEnum

Retrieves or sets a value that indicates whether the control automatically resizes the cell based on the size of the font.

Type	Description
AutoSizeEnum	An AutoSizeEnum expression that indicates whether the control automatically resizes the cell based on the size of the font.

By default, the AutoSize property is exFontSize. The Font property specifies the control's font. The MaxMonthX property specifies the number of months that can displayed on the horizontal axis. The MaxMonthY property specifies the number of months that can displayed on the vertical axis. Use the AutoSize, FixedCellHeight and FixedCellWidth properties to defines the size to display a day in the calendar control. The FixedCellHeight and FixedCellWidth properties has effect only if the AutoSize is exFixedSize.

The following sample fixes the day's size:

```
Calendar1.AutoSize = exFixedSize
Calendar1.FixedCellHeight = 18
Calendar1.FixedCellWidth = 32
```

property Calendar.BackColor as Color

Retrieves or sets the control's background color.

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

Use the ForeColor property to change the foreground color. Use the BackColor property of Event object to change the cell's background color. Use the Picture property to display a picture on the control's background. The PictureDisplay property retrieves or sets a value that indicates the way how the graphic is displayed on the control's background

property Calendar.Background(Part as BackgroundPartEnum) as Color

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

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

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

The following VB sample changes the visual appearance for the header of months. The sample uses the "" skin.

```
With Calendar1
   With .VisualAppearance
      .Add &H24, "D:\Temp\ExCalendar.Help\green.ebn"
   End With
   .Background(exDateHeader) = &H24000000
End With
```

The following C++ sample changes the visual appearance for the header of months:

```
m_calendar.GetVisualAppearance().Add( 0x24,
```

```
COleVariant("D:\\Temp\\ExCalendar.Help\\green.ebn"));
m_calendar.SetBackground( 8 /*exDateHeader*/, 0x24000000 );
```

The following VB.NET sample changes the visual appearance for the header of months:

```
With AxCalendar1
   With .VisualAppearance
      .Add(&H24, "D:\Temp\ExCalendar.Help\green.ebn")
   End With
   .set_Background(EXCALENDARLib.BackgroundPartEnum.exDateHeader, &H24000000)
End With
```

The following C# sample changes the visual appearance for the header of months:

```
axCalendar1.VisualAppearance.Add(0x24, "D:\\Temp\\ExCalendar.Help\\green.ebn");
axCalendar1.set_Background(EXCALENDARLib.BackgroundPartEnum.exDateHeader,
0x24000000);
```

The following VFP sample changes the visual appearance for the header of months:

```
With thisform.Calendar1
   With .VisualAppearance
      .Add(36, "D:\Temp\ExCalendar.Help\green.ebn")
   EndWith
   .Background(8) = 603979776     && exDateHeader
EndWith
```

where the 603979776 is 0x24000000 in hexa, and 36 is 0x24 in hexa

method Calendar.BeginUpdate ()

Maintains performance while multiple changes are done at once. This method prevents the control from painting until the EndUpdate method is called.

Type	Description

Use the BeginUpdate/EndUpdate methods to prevent control's updating while you are performing multiple changes into the control. The Refresh method refreshes the control's content.

property Calendar.BorderLineColor as Color

Retrieves or sets a value that indicates the month's border line color.

Type	Description
Color	A color expression that indicates the month's border line color.

Use the Appearance property to change the control's border. Use the DrawBorderLine property to define the style of month's border line, or to hide it.

property Calendar.CommentBackColor as Color

Retrieves or sets the color to highlight the commented events.

Type	Description
Color	A Color expression that specifies the color to mark the dates that have a comment or a tooltip assigned, 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 for dates with comments assigned

The CommentBackColor property indicates the color to mark the dates with the Comment property being set. If the CommentBackColor property is identical with the BackColor property, the dates are not marked as they have assigned a tooltip or a comment.

property Calendar.Date as Date

Retrieves or sets the browsed date.

Type	Description
Date	A DATE expression that indicates the browsed date.

Ensures that the date is visible. Use the SelDate property to select a date. When the browsed date is changed the control fires the DateChanged event. By default, the Date property points to the current date. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

Use the VB Date function in order to get the current date. use the DoDate property to build a DATE expression given year, month and day. The following sample shows how to browse the month "January 2000":

```
Calendar1.Date = Calendar1.DoDate(2000, 1, 1)
```

property Calendar.DateFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Date

Retrieves the date from point.

Type	Description
X as OLE_XPOS_PIXELS	A single expression that indicates the X position in client coordinate
Y as OLE_YPOS_PIXELS	A single expression that indicates the Y position in client coordinate
Date	0 or a DATE expression that indicates the date from point (X,Y)

Use the DateFromPoint property to get the date from the cursor. if X = -1 and Y = -1, the DateFromPoint property retrieves the date from the cursor, shortly the DateFromPoint(-1,-1) returns the date from the cursor.

The following VB sample displays the date being clicked:

```
Private Sub Calendar1_Click()
   Dim d As Date
   With Calendar1
      d = .DateFromPoint(-1, -1)
      If Not (d = 0) Then
         MsgBox "You have clicked: " & d
      End If
   End With
End Sub
```

The following VB sample displays the date being clicked, including the associated event:

```
Private Sub Calendar1_Click()
   Dim d As Date, s As String
   With Calendar1.Object
      d = .DateFromPoint(-1, -1)
      If Not (d = 0) Then
         s = "You have clicked: " & d
         Dim e As EXCALENDARLibCtl.Event
         Set e = .Events.Item(CDate(d))
```

```
        If Not e Is Nothing Then
            s = s + vbCrLf + "The date has associated an event"
        End If
    End If
  End With
  If (Len(s) > 0) Then
    MsgBox s
  End If
End Sub
```

In VBA/MSAccess, you need to replace the EXCALENDARLibCtl with EXCALENDARLib, else you will be prompted for a compiler error: "Can't find project or library"

The following sample shows how to print the date over the cursor:

```
Private Sub Calendar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Dim d As Date
  d = Calendar1.DateFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
  If d <> 0 Then
    Debug.Print FormatDateTime(d)
  End If
End Sub
```

property Calendar.DateHeaderField(Field as HeaderFieldEnum) as String

Specifies the HTML caption to be shown on the giving field of the date's header.

Type	Description
Field as HeaderFieldEnum	A HeaderFieldEnum expression that defines the index of the part to be changed.
String	A string expression that specifies the HTML caption to be shown on the giving field of the date's header. The DateHeaderField property could be also an expression whose result, can be the HTML caption to be shown on the giving field of the date's header. The **<%month%>** specifies the name of the browsed month, while the **<%year%>** defines the year of the browsed month

By default, the DateHeaderField property for:

- exHeaderDate is "<c><%month%> <%year%>"
- exHeaderPrevMonth is "<c><sha ;;0>-"
- exHeaderNextMonth is "<c><sha ;;0>+"
- exHeaderPrevYear is "<c><sha ;;0><"
- exHeaderNextYear is "<c><sha ;;0>>"

The DateHeaderFormat property specifies the CRD format to display the month/year/buttons within the date's header. The DateHeaderField property has effect only if the DateHeaderFormat property is not empty. The DateHeaderField property could be also an expression that returns the HTML caption to be displayed on the giving field of the date's header. In other words, if the expression of the DateHeaderField property is not valid, it indicates directly the HTML caption, else the HTML caption is the result of evaluating the giving expression.

For instance:

- "<c><%month%> <%year%>", shows the month and the year centered
- "<c><img>up</img>", displays the image named up centered. Previously, the HTMLPicture("up") should be called to specify the location of the picture named up.
- "`<c><%month%> (` + (dateF(value) left 2) + `) <r><off -6><font ;6><%year%>`" includes the month number in the date's header
- " (month(value) = month(date(``)) ? `<b><fgcolor=0000FF>` : ``) + `<c><%month%> <%year%>`" shows the current month with a different foreground color (bold, blue)

The DateHeaderField property supports the following HTML tags:

- ** ... ** displays the text in **bold**

- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The 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:

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

The expression of the DateHeaderField property could use keywords such as:

- **value** which indicates the date of the month to be formatted

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 Calendar.DateHeaderFormat as String

Specifies the CRD format to display the month/year/buttons within the date's header.

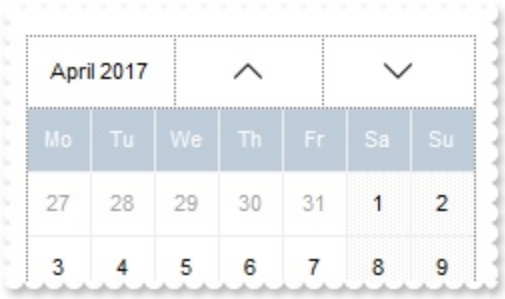
Type	Description
String	A String expression that defines the CRD format to display the month/year/buttons within the date's header. The value of the DateHeaderFormat property could be also an expression that returns a CRD format. You can use the eXCRD tool to define, edit and view CRD strings.

By default, the DateHeaderFormat property is "", which indicates that it has no effect. The DateHeaderFormat property helps you to customize the date's header which includes the month, year and the prev / next month/year buttons. The DateHeaderFormat property could be also an expression that returns a CRD format for specified month. In other words, if the expression of the DateHeaderFormat property is not valid, it indicates directly the CRD syntax, else the CRD syntax is the result of evaluating the giving expression. The DateHeaderField property specifies the HTML caption to be shown on the giving field of the date's header.

The following screen show shows the control with a different date header:

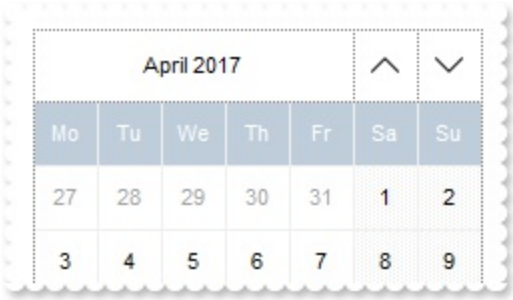
For instance, here are few simple CRD strings:

- The CRD string "`**1,2,3**`" divides the header in three parts, the left side displays the month/year, the middle part displays the next month button, while the last part displays the next month button. Similar with horizontally splitting a cell in three pieces.



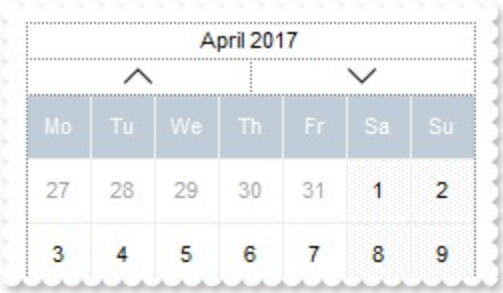
- The CRD string "`**1,2:32,3:32**`" divides the header in three parts, the left side displays the month/year, the middle part displays the next month button, while the last part

displays the next month button. The last two-parts of the CRD are 32-pixels wide.



- The CRD string "`**1,(2/3):32**`" splits horizontally the header in two parts, where the left part displays the month/year while the right-part of 32-pixels wide is divided in two parts, the upper part displays the prev month button and the bottom part displays the next month button.

- The CRD string "`**1/2,3**`" splits the header in two, the upper part displays the month/year, the bottom part is divided in other two parts, where the left part displays the prev month button, and the right part displays the next month button.



You can use the eXCRD tool to define, edit and view CRD strings.

The DateHeaderFormat property can include any of the following:

- **1** index in the CRD format represents the month/year. The DateHeaderField(exHeaderDate) / DateHeaderField(1) property defines what an 1-index part of the CRD string displays. By default, it displays the month and the year of the browsed date. Clicking the 1-index part shows the control's months selector, so the user can select a different month/year.
- **2** specifies the button to go previously one month. The DateHeaderField(exHeaderPrevMonth) / DateHeaderField(2) property defines what an 2-index part of the CRD string displays. By default, it displays "-" character. Clicking

the 2-index part navigates the control to previously month.
- **3** specifies the button to advance to the next month. The DateHeaderField(exHeaderNextMonth) / DateHeaderField(3) property defines what an 3-index part of the CRD string displays. By default, it displays "+" character. Clicking the 3-index part navigates the control to next month.
- 4 specifies the button to go previously one year. The DateHeaderField(exHeaderPrevYear) / DateHeaderField(4) property defines what an 4-index part of the CRD string displays. By default, it displays "<" character. Clicking the 4-index part navigates the control to previously year.
- **5** specifies the button to advance one year. The DateHeaderField(exHeaderNextYear) / DateHeaderField(5) property defines what an 5-index part of the CRD string displays. By default, it displays ">" character. Clicking the 5-index part navigates the control to next year.

For instance, DateHeaderFormat property on

- "1,2:24,3:24" displays the month/year(1), and aligned to the right with a 24-pixels wide the prev(2) and next(3) month-buttons.
- "month(value) = 1 ? `4:24,5:24,1,2:24,3:24` : `1,2:24,3:24`", specifies for January month to include all buttons, and for the rest just the prev and next month-buttons.

The expression of the DateHeaderFormat property could use keywords such as:

- **value** which indicates the date of the month to be formatted
- **x** indicates the x-position of the month within the calendar
- **xmax** specifies the number of months being displayed horizontally in the calendar
- **y** indicates the y-position of the month within the calendar
- **ymax** specifies the number of months being displayed vertically in the calendar.

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 Calendar.DisplayWeekNumberAs as WeekNumberAsEnum

Specifies the way the control displays the week number.

Type	Description
WeekNumberAsEnum	A WeekNumberAsEnum expression that specifies the ways the control displays the week number for dates.

By default, the DisplayWeekNumberAs property is exISO8601WeekNumber, which indicates that the week number is displayed according to the ISO8601 standard, which specifies that the first week of the year is the one that includes the January the 4th. The ShowWeeks property specifies whether the week number header is shown or hidden. The FirstDay property specifies the first day of the week where the week begins.

The following screen show shows the calendar while using the DisplayWeekNumberAs property on exISO8601WeekNumber (default):

The following screen show shows the calendar while using the DisplayWeekNumberAs property on exSimpleWeekNumber:

property Calendar.DoDate (Year as Long, Month as Long, Day as Long) as Date

Composes a DATE type, based on year, month and day.

Type	Description
Year as Long	A long expression that indicates the year.
Month as Long	A long expression that indicates the month. 1 - January, 2 - February, ... 12 - December
Day as Long	A long expression that indicates the day number. 1 - First day of the month, ... If the Day is -1, the DoDate returns the last day of the specified month/year.
Date	A DATE expression that indicates the built date

Use the DoDate to build a DATE expression given the year, month and the day. If the Day is -1, the DoDate returns the last day of the specified month/year.

property Calendar.DrawBorderLine as LineStyleEnum

Retrieves or sets a value that indicates the month's border line style.

Type	Description
LineStyleEnum	A LineStyleEnum expression that indicates the month's border line style.

Use the BorderLineColor property to change the color for the month's border. Use the DrawBorderLine to hide the month's border line.

property Calendar.DrawGridLine as LineStyleEnum

Retrieves or sets a value that identifies the type of grid lines.

Type	Description
LineStyleEnum	A LineStyleEnum expression that indicates whether the grid lines are visible or hidden.

Use the DrawGridLine property to show the grid lines. Use the GridLineColor property to define the color for grid lines.

property Calendar.Enabled as Boolean

Retrieves or sets a value that indicates whether the control is enabled or disabled.

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

Use the Enabled property to disable the control. Use the Locked property to lock the control's selection.

method Calendar.EndUpdate ()

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

Type	Description

Use the BeginUpdate/EndUpdate methods to prevent control's updating while you are performing multiple changes into the control. The Refresh method refreshes the control's content.

property Calendar.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. If -2, the EventParam gives full information about the event, such as name, identifier, and 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 Calendar.Events as Events

Retrieves the control date events collection.

Type	Description
Events	An Events collection that contains Event objects. Each Event defines a DATE and its graphical information like: foreground color, background color, font attributes, tooltips, images and so on.

Use the Events property to access the Event objects.

The following sample applies a bold effect to "yesterday" date:

```
Calendar1.Events.Add(Date - 1).Bold = True
```

method Calendar.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 number of events within the control:

```
Debug.Print Calendar1.ExecuteTemplate("Events.Count")
```

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 Calendar.FirstDay as WeekDayEnum

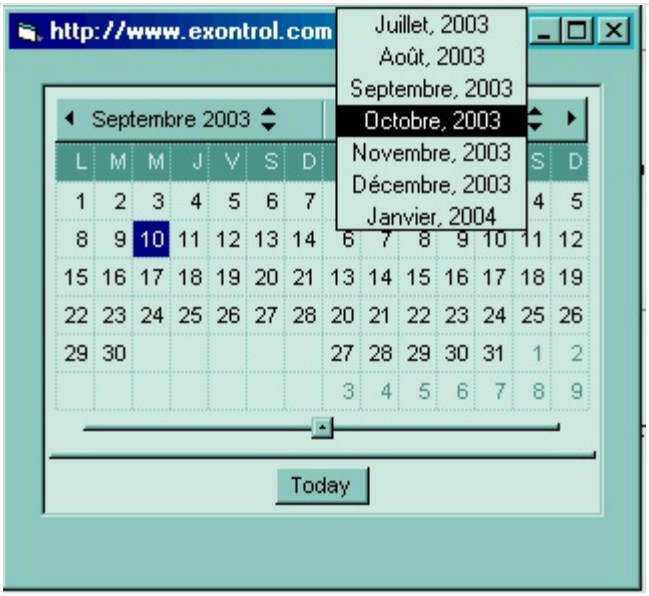
Retrieves or sets a value that indicates the first day of the week.

Type	Description
WeekDayEnum	A WeekDayEnum expression that defines the first day of the week.

By default, the FirstDay property is Sunday. Use the FirstDay property to define the month for a specific language. Use the WeekDays property to define the shortcut for each week day. Use the MonthNames to define the name for each month. For instance, in Europe, the week starts on Monday. The DisplayWeekNumberAs property specifies the way the control displays the week number. Use the ShowWeeks property to display the week numbers. The LocFirstDay property indicates the first day of the week, using the current user regional and language settings.

The following sample defines the "French" style:

```
Private Sub Form_Load()
   With Calendar1
      .FirstDay = Monday
      .MonthNames = "Janvier Février Mars Avril Mai Juin Juillet Août Septembre Octobre
Novembre Décembre"
      .WeekDays = "D L M M J V S"
   End With
End Sub
```



property Calendar.FirstVisibleDate as Date

Retrieves the first visible date.

Type	Description
Date	A DATE expression that specified the first visible date in the calendar.

The FirstVisibleDate property retrieves the first visible date being displayed in the calendar. Use the ShowNonMonthDays property to display the dates that are not part of the month. Use the LastVisibleDate property to get the last visible date being displayed in the calendar.

property Calendar.FixedCellHeight as Long

Retrieves or sets a value that indicates the cell's height while the AutoSize is false.

Type	Description
Long	A long expression that defines the cell's height while the AutoSize property is False.

Use the AutoSize, FixedCellHeight and FixedCellWidth properties to defines the size of the control's cell. A cell displays a date. The FixedCellHeight and FixedCellWidth properties has effect only if the AutoSize is False.

The following sample fixes the cell's size to (18,32):

```
Private Sub Form_Load()
  With Calendar1
    .AutoSize = False
    .FixedCellHeight = 18
    .FixedCellWidth = 32
  End With
End Sub
```

property Calendar.FixedCellWidth as Long

Retrieves or sets a value that indicates the cell's width while the AutoSize is false.

Type	Description
Long	A long expression that defines the cell's width while the AutoSize property is False.

Use the AutoSize, FixedCellHeight and FixedCellWidth properties to defines the size of the control's cell. A cell displays a date. The FixedCellHeight and FixedCellWidth properties has effect only if the AutoSize is False.

The following sample fixes the cell's size:

```
Private Sub Form_Load()
   With Calendar1
      .AutoSize = False
      .FixedCellHeight = 18
      .FixedCellWidth = 32
   End With
End Sub
```

property Calendar.FocusDate as Date

Retrieves or sets the focused date

Type	Description
Date	A DATE expression that indicates the date being focused.

Use the FocusDate property to get the date that has the focus. Use the SingleSel property to specify whether the control supports single or multiple selection. Use the HideSelection property to specify whether the selection is hidden when the control loses the focus. Use the FocusDate property to select a new date, if the control's SingleSel property is True. The control fires the FocusChanged event when a new date is focused. If the control's SingleSel property is False or the date that has the focus is disabled, the control draws a focus rectangle around the date. Use the Disabled property to disable a date. Use the SelDate and SelCount properties to retrieve or sets the collection of selected dates. If the new focused date requires browsing a new date, the control fires the DateChanged event.

property Calendar.Font as IFontDisp

Retrieves or sets the font to display the drop down portion of the control.

Type	Description
IFontDisp	A Font object that defines the control's font.

Defines the drop down portion's s font. Use the Bold, Italic, UnderLine or StrikeOut property to change the font attributes for a particular date.

property Calendar.ForeColor as Color

Retrieves or sets the control's foreground color.

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

Defines the control's foreground color. Use the ForeColor property of Event object to define the foreground color for a particular date.

method Calendar.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 Calendar.

The FormatABC method formats the A,B,C values based on the giving expression and returns the result.

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 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

This property/method supports predefined constants and operators/functions as described here.

method Calendar.FreezeEvents (Freeze as Boolean)

Prevents the control to fire any event.

Type	Description
Freeze as Boolean	A Boolean expression that specifies whether the control' events are froze or unfroze

The FreezeEvents(True) method freezes the control's events until the FreezeEvents(False) method is called.

property Calendar.GridLineColor as Color

Retrieves or sets a value that indicates the grid lines color.

Type	Description
Color	A color expression that indicates the grid lines color.

Use the DrawGridLine property to hide the grid lines.

property Calendar.HeaderBackColor as Color

Retrieves or sets a value that indicates the background color used for weeks and week days headers.

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

Use the HeaderForeColor property to change the foreground color for weeks and week days headers. Use the ShowWeeks to hide the weeks header. Use the ShowDays to hide the week days header. Use the Background property to change the visual appearance for parts in the control.

property Calendar.HeaderForeColor as Color

Retrieves or sets a value that indicates the foreground color used for weeks and week days headers.

Type	Description
Color	A color expression that indicates the week and week days's foreground color

Use the HeaderBackColor property to change the foreground color for weeks and week days headers. Use the ShowWeeks to hide the weeks header. Use the ShowDays to hide the week days header.

property Calendar.HideSelection as Boolean

Specifies whether selected date appears highlighted when a control loses focus.

Type	Description
Boolean	A boolean expression that specifies whether selected date appears "highlighted" when a control loses focus.

By default, the HideSelection property is False.

property Calendar.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

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

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, tooltips, comments, 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).

property Calendar.hWnd as Long

Retrieves the control's window handle.

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

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

method Calendar.Images (Handle as Variant)

Sets the control's handle image list.

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 the control images panel. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ShowImageList property to hide the control's images panel. Use the Image property to assign an icon to a date.

The following sample loads icons from a BASE64 encoded strings:

```
With Calendar1
   .BackColor = vbWhite
   .AutoSize = False
   .FixedCellWidth = 32
   .Images
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExm

   With .Events.Add(Date + 1)
      .Image = 1
   End With
   With .Events.Add(Date + 2)
      .Image = 2
   End With
End With
```

If you run the sample you will see:

The following sample uses the Microsoft Image List control:

```
Calendar1.Images ImageList1.hImageList
```

The following screen shot shows the control's images panel:

Calendar1
Images
Drag here an executable file,
a dynamic-link library, an
icon file or a regular file

property Calendar.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 Calendar.IntegralHeight as Long

Retrieves the height of the control to fit the MaxMonthY months in the client area.

Type	Description
Long	A long expression that indicates the height in pixels that control requires in order to display a number of MaxMonthY months in the client area.

Use the MaxMonthY property to specify the maximum number of months displayed in the client area. Use the IntegralWidth property to retrieve the required width for the control to fit a number of MaxMonthX months in the client area. The following sample changes the control's size to let control displays 2 x 2 months in the client area:

```
With Calendar1
    .MaxMonthX = 2
    .MaxMonthY = 2
    .Width = Screen.TwipsPerPixelX * .IntegralWidth
    .Height = Screen.TwipsPerPixelY * .IntegralHeight
End With
```

property Calendar.IntegralWidth as Long

Retrieves the width of the control to fit the MaxMonthX months in the client area.

Type	Description
Long	A long expression that indicates the width in pixels that control requires in order to display a number of MaxMonthX months in the client area.

Use the MaxMonthX property to specify the maximum number of months displayed in the client area. Use the IntegralHeight property to retrieve the required height for the control to fit a number of MaxMonthY months in the client area. The following sample changes the control's size to let control displays 2 x 2 months in the client area:

```
With Calendar1
    .MaxMonthX = 2
    .MaxMonthY = 2
    .Width = Screen.TwipsPerPixelX * .IntegralWidth
    .Height = Screen.TwipsPerPixelY * .IntegralHeight
End With
```

property Calendar.LastVisibleDate as Date

Retrieves the last visible date.

Type	Description
Date	A DATE expression that specified the last visible date in the calendar.

The LastVisibleDate property retrieves the last visible date being displayed in the calendar. Use the ShowNonMonthDays property to display the dates that are not part of the month. Use the FirstVisibleDate property to get the first visible date being displayed in the calendar.

property Calendar.LocAMPM ([Abbreviation as Variant]) as String

Retrieves the time marker such as AM or PM using the current user regional and language settings.

Type	Description
Abbreviation as Variant	An optional parameter that indicates the number of characters to be retrieved. If missing or -1, the entire field is retrieved.
String	A String expression that indicates the time marker such as AM or PM using the current user regional and language settings.

The LocAMPM property gets the locale AM/PM indicators as indicated by current regional settings. The LocFirstDay property indicates the first day of the week, using the current user regional and language settings. The LocMonthNames property specifies the list of name of the months, using the current user regional and language settings. The LocWeekDays property specifies the name of the days in the week, using the current user regional and language settings.

property Calendar.LocFirstDay as WeekDayEnum

Indicates the first day of the week, as specified in the regional settings.

Type	Description
WeekDayEnum	A WeekDayEnum expression that specifies the first day of the week, as specified in the regional settings.

The LocFirstDay property indicates the first day of the week, using the current user regional and language settings. The LocMonthNames property specifies the list of name of the months, using the current user regional and language settings. The LocWeekDays property specifies the name of the days in the week, using the current user regional and language settings. The LocAMPM property gets the locale AM/PM indicators as indicated by current regional settings.

property Calendar.Locked as Boolean

Specifies whether the user can change the selection.

Type	Description
Boolean	A boolean expression that specifies whether the user can change the selection.

Use the Locked property to lock the control. If the control is locked the user can scroll the control's content. Use the Enabled property to disable the control. If the control is disabled the control cannot scroll the control's content.

property Calendar.LocMonthNames ([Abbreviation as Variant]) as String

Retrieves the list of month names, as indicated in the regional settings, separated by space.

Type	Description
Abbreviation as Variant	An optional parameter that indicates the number of characters to be retrieved. If missing or -1, the entire field is retrieved.
String	A String expression that indicates the name of the months within the year, as indicated in the regional settings, separated by space.

Use the LocMonthNames property to get the name of the months as indicated by current regional settings. The LocFirstDay property indicates the first day of the week, as indicated in the regional settings. The LocAMPM property specifies specifies the AM and PM indicators, as indicated in the regional settings. The LocWeekDays property specifies the name of the days in the week, as indicated in the regional settings.

property Calendar.LocWeekDays ([Abbreviation as Variant]) as String

Retrieves the list of names for each week day, as indicated in the regional settings, separated by space.

Type	Description
Abbreviation as Variant	An optional parameter that indicates the number of characters to be retrieved. If missing or 1, only the first character of the week day is retrieved.
String	A String expression that indicates the list of names for each week day, as indicated in the regional settings, separated by space.

The LocWeekDays property specifies the name of the days in the week, using the current user regional and language settings. The LocAMPM property gets the locale AM/PM indicators as indicated by current regional settings. The LocFirstDay property indicates the first day of the week, using the current user regional and language settings. The LocMonthNames property specifies the list of name of the months, using the current user regional and language settings.

property Calendar.MarkToday as Boolean

Retrieves or sets a value that indicates whether the control marks the today date.

Type	Description
Boolean	A boolean expression that indicates whether the control marks the today date.

Use the MarkToday property to mark today date. By default, the MarkToday property is False. Use the Background property to change the visual appearance for parts in the control.

property Calendar.MaxDate as Date

Retrieves or sets the min date.

Type	Description
Date	A Date expression that indicates the upper limit for dates to be displayed or selected.

By default, the MaxDate property is Dec 31, 9999. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected. The Date property is automatically updated so the browsed date fits the limited range. When the browsed date is changed the control fires the DateChanged event. By default, the Date property points to the current date.

property Calendar.MaxMonthX as Long

Specifies the maximum number of months horizontally displayed.

Type	Description
Long	A long expression that specifies the maximum number of months horizontally displayed.

By default, the MaxMonthX property is 6. Use the MaxMonthX property to define the maximum number of months displayed horizontally. Use the MinMonthX property to define the minimum number of months displayed horizontally. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property Calendar.MaxMonthY as Long

Specifies the maximum number of months vertically displayed.

Type	Description
Long	A long expression that defines the maximum number of months vertically displayed.

By default, the MaxMonthY property is 2. Use the MaxMonthY property to define the maximum number of months vertically displayed. Use the MinMonthY property to define the minimum number of months vertically displayed. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property Calendar.MaxScrollYear as Long

Specifies the maximum year when scrolling.

Type	Description
Long	A long expression that indicates the maximum year when scrolling.

Use the MinScrollYear and MaxScrollYear properties to specify the range of dates for your calendar control.

property Calendar.MinDate as Date

Retrieves or sets the min date.

Type	Description
Date	A Date expression that indicates the lower limit for dates to be displayed or selected.

By default, the MinDate property is Jan 1, 100. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected. The Date property is automatically updated so the browsed date fits the limited range. When the browsed date is changed the control fires the DateChanged event. By default, the Date property points to the current date.

property Calendar.MinMonthX as Long

Specifies the minimum number of months horizontally displayed.

Type	Description
Long	A long expression that specifies the minimum number of months horizontally displayed.

By default, the MinMonthX property is 1. Use the MinMonthX property to define the minimum number of months displayed horizontally. Use the MaxMonthX property to define the maximum number of months displayed horizontally. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property Calendar.MinMonthY as Long

Specifies the minimum number of months vertically displayed.

Type	Description
Long	A long expression that specifies the minimum number of months vertically displayed.

By default, the MinMonthY property is 1. Use the MinMonthY property to define the minimum number of months vertically displayed. Use the MaxMonthY property to define the maximum number of months vertically displayed. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property Calendar.MinScrollYear as Long

Specifies the minimum year when scrolling.

Type	Description
Long	A long expression that indicates the minimum year when scrolling.

Use the MinScrollYear and MaxScrollYear properties to specify the range of dates for your calendar control.

property Calendar.MonthName(Month as MonthEnum) as String

Retrieves or sets the month's name.

Type	Description
Month as MonthEnum	A MonthEnum expression that indicates the month
String	A String expression that specifies the month's name.

Use the MonthNames property to change the names for all months.

property Calendar.MonthNames as String

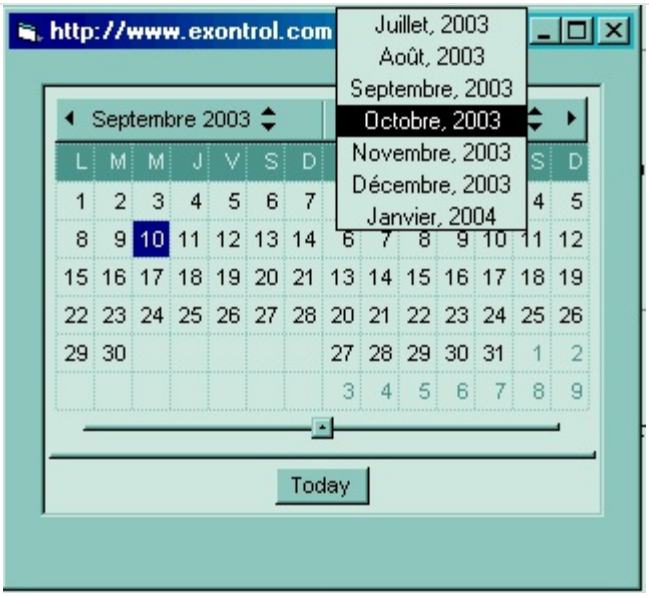
Retrieves or sets a value that indicates the list of month names, separated by space.

Type	Description
String	A string expression hat indicates the list of month names, separated by space.

By default, the MonthNames is "January February March April May June July August September October November December". For instance, for French you have to use somethign like: "Janvier Février Mars Avril Mai Juin Juillet Août Septembre Octobre Novembre Décembre". Use the FirstDay property to change the first day of the week. Use the WeekNames property to define the name for each week day. LocMonthNames property specifies the list of name of the months, using the current user regional and language settings.

The following sample defines the "French" style:

```
Private Sub Form_Load()
   With Calendar1
      .FirstDay = Monday
      .MonthNames = "Janvier Février Mars Avril Mai Juin Juillet Août Septembre Octobre
Novembre Décembre"
      .WeekDays = "D L M M J V S"
   End With
End Sub
```



property Calendar.NonMonthDaysColor as Color

Retrieves or sets a value that indicates the color to show the non-month days.

Type	Description
Color	A Color expression that specifies the color to show the days that are not owned by the current month.

By default, the NonMonthDaysColor property is gray. The NonMonthDaysColor property specifies the foreground color to show the days that are not owned by displaying month. The ShowNonMonthDays property specifies whether the calendar displays the days that are not-owned by the displaying month. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date. The NonworkingDaysForeColor property specifies the foreground color for non-working days.

The following screen shot shows the non-month days in red:

property Calendar.NonworkingDays as Long

Retrieves or sets a value that indicates the non-working days, for each week day a bit.

Type	Description
Long	A long expression that indicates the non-working days in a week.

By default, the NonworkingDays property is 65. The last significant byte in the NonworkingDays expression has the following meaning:

-	Sa	Fr	Th	We	Tu	Mo	Su
0	X	X	X	X	X	X	X
128	64	32	16	8	4	2	1

where **X** could be 1 (nonworking day) or 0 (working day), **Sa** means Saturday, **Fr** means Friday, and so on. For instance, the 65 value means Saturday and Sunday are non-working days. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days. For instance, if the NonworkingDaysPattern is exPatternEmpty the non-working days are not highlighted. Use the ShowNonMonthDays property to specify whether the dates that are not part of the month are visible or hidden. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date. The FirstDay property specifies the first day of the week. The NonworkingDaysForeColor property specifies the foreground color for non-working days.

The following VB sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
With Calendar1
   .NonworkingDays = 2 ^ (EXCALENDARLibCtl.Sunday - 1) Or 2 ^
(EXCALENDARLibCtl.Monday - 1)
End With
```

The following C++ sample retrieves the value to indicate Sunday and Monday as being non-

working days:

```
m_calendar.SetNonworkingDays( 1 << ( EXCALENDARLib::Sunday - 1 ) | 1 << (
EXCALENDARLib::Monday - 1 ) );
```

The following VB.NET sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
With AxCalendar1
    .NonworkingDays = 2 ^ (EXCALENDARLib.WeekDayEnum.Sunday - 1) Or 2 ^
(EXCALENDARLib.WeekDayEnum.Monday - 1)
End With
```

The following C# sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
axCalendar1.NonworkingDays = 1 <<
(Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Sunday) - 1) | 1 <<
(Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Monday) - 1);
```

The following VFP sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
with thisform.Calendar1
    .NonworkingDays = 2 ^ 0 + 2 ^ 1
endwith
```

property Calendar.NonworkingDaysColor as Color

Retrieves or sets a value that indicates the color to fill the non-working days.

Type	Description
Color	A Color expression that indicates the color to fill the non-working days.

Use the NonworkingDaysColor property to specify the color being used by the NonworkingDaysPattern property. Use the NonworkingDays property to specify the nonworking days in a week. Use the NonworkingDaysPattern property to specify the pattern to fill the non-working days. For instance, if the NonworkingDaysPattern is exPatternEmpty the non-working days are not highlighted. The FirstDay property specifies the first day of the week. The NonworkingDaysForeColor property specifies the foreground color for non-working days.

The following VB sample marks Sunday and Monday days on red:

```
With Calendar1
   .NonworkingDays = 2 ^ (EXCALENDARLibCtl.Sunday - 1) Or 2 ^
(EXCALENDARLibCtl.Monday - 1)
   .NonworkingDaysColor = RGB(255, 0, 0)
End With
```

The following C++ sample sample marks Sunday and Monday days on red:

```
m_calendar.SetNonworkingDays( 1 << ( EXCALENDARLib::Sunday - 1 ) | 1 << (
EXCALENDARLib::Monday - 1 ) );
m_calendar.SetNonworkingDaysColor( RGB(255,0,0,) );
```

The following VB.NET sample marks Sunday and Monday days on red:

```
With AxCalendar1
   .NonworkingDays = 2 ^ (EXCALENDARLib.WeekDayEnum.Sunday - 1) Or 2 ^
(EXCALENDARLib.WeekDayEnum.Monday - 1)
   .NonworkingDaysColor = Color.Red
End With
```

The following C# sample marks Sunday and Monday days on red:

```
axCalendar1.NonworkingDays = 1 <<
```

```
(Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Sunday) - 1) | 1 <<
(Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Monday) - 1);
axCalendar1.NonworkingDaysColor = Color.Red;
```

The following VFP sample sample marks Sunday and Monday days on red:

```
with thisform.Calendar1
   .NonworkingDays = 2 ^ 0 + 2 ^ 1
   .NonworkingDaysColor = RGB(255,0,0)
endwith
```

property Calendar.NonworkingDaysForeColor as Color

Retrieves or sets a value that indicates the foreground color for non-working days.

Type	Description
Color	A Color expression that specifies the color to show the non-working days of the month.

By default, the NonworkingDaysForeColor property is black. The NonworkingDaysForeColor property specifies the foreground color for non-working days. The NonworkingDaysColor property specifies the color to show the pattern for non-working days. Use the NonworkingDays property to specify the nonworking days in a week. The FirstDay property specifies the first day of the week. Use the NonworkingDaysPattern property to specify the pattern to fill the non-working days. Use the NonMonthDaysColor property to specify the color to show days that are not owned by displayed month. By default, the ForeColor property specifies the color to show the days.

The following screen shot shows the non-working days in red:

property Calendar.NonworkingDaysPattern as PatternEnum

Retrieves or sets a value that indicates the pattern being used to fill non-working days.

Type	Description
PatternEnum	A PatternEnum expression that indicates the pattern to fill non working days.

Use the NonworkingDaysPattern property to specify the pattern to fill non-working days. By default, the NonworkingDaysPattern property is exPatternDot. If the NonworkingDaysPattern property is exPatternEmpty, the non-working days are not highlighted. Use the NonworkingDays property to specify the non-working days in a week. The NonworkingDaysColor property specifies the color being used to fill the non-working days. The NonworkingDaysForeColor property specifies the foreground color for non-working days. The FirstDay property specifies the first day of the week.

The following VB sample draws non-working days using the exPatternShadow brush:

```
With Calendar1
   .NonworkingDaysPattern = exPatternShadow
End With
```

The following C++ sample draws non-working days using the exPatternShadow brush:

```
m_calendar.SetNonworkingDaysPattern( 3 /*exPatternShadow*/ );
```

The following VB.NET sample draws non-working days using the exPatternShadow brush:

```
With AxCalendar1
   .NonworkingDaysPattern = EXCALENDARLib.PatternEnum.exPatternShadow
End With
```

The following C# sample draws non-working days using the exPatternShadow brush:

```
axCalendar1.NonworkingDaysPattern = EXCALENDARLib.PatternEnum.exPatternShadow
```

The following VFP sample draws non-working days using the exPatternShadow brush:

```
with thisform.Calendar1
    .NonworkingDaysPattern = 3
endwith
```

property Calendar.Picture as IPictureDisp

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

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

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the BackColor property to specify the control's background color. You can use the Picture property to add your logo on the control's background. The /NET version provides the BackgroundImage property.

The following screen shot show a picture on the control's background:

property Calendar.PictureDisplay as PictureDisplayEnum

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

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

By default, the PictureDisplay property is exTile. The PictureDisplay property specifies how the Picture is displayed on the control's background. If the control has no picture associated the PictureDisplay property has no effect. Use the BackColor property to specify the control's background color. The /NET version provides the BackgroundImageLayout property.

method Calendar.Refresh ()

Refreshes the control.

Type	Description

Use the BeginUpdate/EndUpdate methods to prevent control's updating while you are performing multiple changes into the control.

property Calendar.SelBackColor as Color

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

Type	Description
Color	A color expression that specifies the selection 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.

Use the SelBackColor and SelForeColor properties to define the colors for selected date(s).

The following VB sample changes the visual appearance for the selected date. The SelBackColor property indicates the selection background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the SelBackColor property to indicates the index of the skin that we want to use. The sample applies the "" to the selected date(s):


```
With Calendar1
   With .VisualAppearance
      .Add &H23, "D:\Temp\ExCalendar.Help\seldate.ebn"
   End With
   .SelBackColor = &H23000000
End With
```

The following C++ sample changes the visual appearance for selected date:

```
m_calendar.GetVisualAppearance().Add( 0x23,
COleVariant("D:\\Temp\\ExCalendar.Help\\seldate.ebn"));
m_calendar.SetSelBackColor( 0x23000000 );
```

The following VB.NET sample changes the visual appearance for selected date:

```
With AxCalendar1
   With .VisualAppearance
      .Add(&H23, "D:\Temp\ExCalendar.Help\seldate.ebn")
   End With
   .Template = "SelBackColor = 587202560"
End With
```

where the 587202560 value in hexa representation is &H23000000

The following C# sample changes the visual appearance for selected date:

```
axCalendar1.VisualAppearance.Add(0x23, "D:\\Temp\\ExCalendar.Help\\seldate.ebn");
axCalendar1.Template = "SelBackColor = 587202560";
```

where the 587202560 value in hexa representation is 0x23000000

The following VFP sample changes the visual appearance for selected date:

```
With thisform.Calendar1
   With .VisualAppearance
      .Add(35, "D:\Temp\ExCalendar.Help\seldate.ebn")
   EndWith
   .SelBackColor = 587202560
EndWith
```

where the 587202560 value in hexa representation is 0x23000000, 35 is 0x23 in hexa

property Calendar.SelCount as Long

Retrieves the count of selected dates.

Type	Description
Long	A long expression that indicates the count of selected dates.

Use the SelCount property to enumerate the selected dates when the SingleSel property is False. Use the SelectDate property to get the selected date given its index into selected dates collection. The SelectDate(0) and SelDate properties returns the same result (are equivalents). Use the FocusDate property to specify the date that has the focus. Use the UnSelDate property to unselect a date.

The following VB sample unselects all dates:

```
With Calendar1
   While .SelCount() > 0
      .UnSelDate .SelectDate(0)
   Wend
End With
```

The following C++ sample unselects all dates:

```
while ( m_calendar.GetSelCount() > 0 )
   m_calendar.UnSelDate( m_calendar.GetSelectDate( 0 ) );
```

The following VB.NET sample unselects all dates:

```
With AxCalendar1
   While .SelCount > 0
      .UnSelDate(.get_SelectDate(0))
   End While
End With
```

The following C# sample unselects all dates:

```
while (axCalendar1.SelCount > 0)
   axCalendar1.UnSelDate(axCalendar1.get_SelectDate(0));
```

The following VFP sample unselects all dates:

```
with thisform.Calendar1
   do while ( .SelCount > 0 )
      .UnSelDate( .SelectDate(0) )
   enddo
endwith
```

property Calendar.SelDate as Date

Selects a date while SingleSel property is True.

Type	Description
Date	A DATE expression that indicates the selected date.

Use the SelDate property to select a date while the SingleSel property is True. In this case, the date that has the focus is changed too. Use the FocusDate property to specify the date that has the focus. If the SingleSel property is False, Use the SelectDate and SelCount properties to enumerate the selected dates. The control fires the SelectionChanged event when the user selects a new date. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date.

The following VB sample prints the selected date(s):

```
With Calendar1
   Dim i As Long
   For i = 0 To .SelCount - 1
      Debug.Print FormatDateTime(.SelectDate(i), vbLongDate)
   Next
End With
```

The following C++ sample prints the selected date(s):

```
for ( long i = 0 ; i < m_calendar.GetSelCount() ; i++ )
{
   COleDateTime date = m_calendar.GetSelDate();
   OutputDebugString( date.Format() );
}
```

The following VB.NET sample prints the selected date(s):

```
With AxCalendar1
   Dim i As Integer
   For i = 0 To .SelCount - 1
      System.Diagnostics.Debug.WriteLine(.get_SelectDate(i).ToString())
   Next
End With
```

The following C# sample prints the selected date(s):

```
for (int i = 0; i < axCalendar1.SelCount; i++)
   System.Diagnostics.Debug.WriteLine(axCalendar1.get_SelectDate(0).ToString());
```

The following VFP sample prints the selected date(s):

```
with thisform.Calendar1
   local i, d
   for i = 0 to .SelCount -1
      d = .SelectDate(i)
   next
endwith
```

property Calendar.SelectDate (Index as Long) as Date

Retrieves the selected date, given its index into selected dates collection. Use SelCount in order to get the count of selected dates.

Type	Description
Index as Long	A long expression that indicates the index of the date into selected dates collection.
Date	A DATE expression that indicates the selected date

If the SingleSel property is False, use the SelCount and SelectDate properties to enumerate the selected dates. Use the UnSelDate property to unselect a date. Use the FocusDate property to specify the date that has the focus.

property Calendar.Selection as String

Serializes the selected dates to a string.

Type	Description
String	A string expression that indicates the selected dates.

Use the Selection property to save the selected dates to a string. Use the Selection property to save and load the selected dates.

property Calendar.SelectTodayDate as Boolean

Specifies whether the current date is selected when the user clicks the Today button.

Type	Description
Boolean	A boolean expression that specifies whether the today date is selected when the user clicks the Today button.

By default, the SelectTodayData property is True. The SelectionChanged event notifies your application when a new date is selected in the calendar control. Use the ShowTodayButton property to specify whether the Today button is shown or hidden. Use the TodayCaption property to change the caption for the today button.

property Calendar.SelForeColor as Color

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

Type	Description
Color	A color expression that indicates the selection foreground color.

Use the SelForeColor and SelBackColor properties to define the foreground and background colors of the selected date.

property Calendar.ShowDays as Boolean

Retrieves or sets a value that indicates whether the week days header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the week days header is visible or hidden.

Use the ShowDays property to hide the week days header. Use the HeaderBackColor and HeaderForeColor properties to change the background and foreground colors of the week days header.

property Calendar.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 False. The control's images panel is visible only at design time. The Images method sets the control's handle image list. The ImageSize property defines the size (width/height) of the icons within the control's Images collection.

The following screen shot shows the control's images panel:

property Calendar.ShowMonth as Boolean

Retrieves or sets a value that indicates whether the months header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the month header is visible or hidden.

Use the ShowMonth property to hide the months's header.

property Calendar.ShowMonthSelector as Boolean

Retrieves or sets a value that indicates whether the user is able to select a new month by clicking in the month header.

Type	Description
Boolean	A boolean expression that indicates whether the user is able to select a new month by clicking in the month header.

By default, the ShowMonthSelector property is True. Use the ShowYearSelector property to specify whether the selector for years are visible or hidden. The DateHeaderFormat property specifies the CRD format to display the month/year/buttons within the date's header. The control DateChanging event when a selector is clicked.

- If the ShowMonthSelector and ShowYearSelector property are True, the Left and Right selectors changes the current year to prev or next year
- If the ShowMonthSelector is False and ShowYearSelector property is True, the Left and Right selectors changes the current month to prev or next month

The following screen shot shows the selectors (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the month selector once the user clicks the month header (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the selectors (the left and right selectors changes the current month to previous or next. no month selector is displayed when the user clicks the month header, **ShowMonthSelector(False), ShowYearSelector(True)**)

property Calendar.ShowNonMonthDays as Boolean

Specifies whether the control displays the dates that are not part of the month.

Type	Description
Boolean	A Boolean expression that indicates whether the dates that are not part of the month are visible or hidden.

By default, the ShowNonMonthDays property is True. Use the ShowNonMonthDays property to hide the dates that are not part of the month. Use the NonWorkingDays property to specify the non working days. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date. The NonMonthDaysColor property specifies the foreground color to show the days that are not owned by displaying month.

The following screen shot shows the control when the ShowNonMonthDays property is False:

The following screen shot shows the control when the ShowNonMonthDays property is True:

property Calendar.ShowTodayButton as Boolean

Retrieves or sets a value that indicates whether the today button is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the today button is visible or hidden.

By clicking the Today button, the Date property is set to today date. Use the ShowTodayButton to show or hide the today button. Use the TodayCaption property to change the caption for the today button.

method Calendar.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/ToolTip 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 Calendar.ShowWeeks as Boolean

Retrieves or sets a value that indicates whether the weeks header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the weeks header is visible or hidden.

Use the ShowWeeks property to show the weeks header. The weeks header displays the week number into the year. If the SingleSel is False, by clicking into the weeks header, the entire week is selected. Use the ShowDays property to hide the week days header. Use the HeaderBackColor and HeaderForeColor properties to change the background and foreground colors of the weeks header. The DisplayWeekNumberAs property specifies the way the control displays the week number. Use the ShowNonMonthDays property to specify whether the dates that are not part of the month are visible or hidden.

In [ISO8601], the week number is defined by:

- weeks start on a monday
- week 1 of a given year is the one that includes the first Thursday of that year. (or, equivalently, week 1 is the week that includes 4 January.)

property Calendar.ShowYearScroll as Boolean

Retrieves or sets a value that indicates whether the scroll bar for changing the year is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the scroll bar for changing the year is visible or hidden

By default, the ShowYearScroll property is True.

property Calendar.ShowYearSelector as Boolean

Retrieves or sets a value that indicates whether the year selector is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the year selector is visible or hidden.

By default, the ShowYearSelector property is True. Use the ShowMonthSelector property to specify whether the selector for month are visible or hidden. The DateHeaderFormat property specifies the CRD format to display the month/year/buttons within the date's header. The control DateChanging event when a selector is clicked.

- If the ShowMonthSelector and ShowYearSelector property are True, the Left and Right selectors changes the current year to prev or next year
- If the ShowMonthSelector is False and ShowYearSelector property is True, the Left and Right selectors changes the current month to prev or next month

The following screen shot shows the selectors (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the month selector once the user clicks the month header (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the selectors (the left and right selectors changes the current month to previous or next. no month selector is displayed when the user clicks the month header, **ShowMonthSelector(False), ShowYearSelector(True)**)

property Calendar.SingleSel as Boolean

Retrieves or sets a value that indicates whether the control supports single or multiple selection.

Type	Description
Boolean	A boolean expression that indicates whether the control supports single or multiple selection.

Use the SingleSel property to specify whether the control supports single or multiple selection. By default, the SingleSel property is True. If the user selects a new date, the SelectionChanged event is fired. Use the SelDate or SelectDate property to get the selected date(s). Use the FocusDate property to specify the date that has the focus. Use the SelCount property to retrieve the number of selected dates. Use the UnSelDate property unselects a date.

The following VB sample prints the selected date(s):

```
With Calendar1
   Dim i As Long
   For i = 0 To .SelCount - 1
      Debug.Print FormatDateTime(.SelectDate(i), vbLongDate)
   Next
End With
```

The following C++ sample prints the selected date(s):

```
for ( long i = 0 ; i < m_calendar.GetSelCount() ; i++ )
{
   COleDateTime date = m_calendar.GetSelDate();
   OutputDebugString( date.Format() );
}
```

The following VB.NET sample prints the selected date(s):

```
With AxCalendar1
   Dim i As Integer
   For i = 0 To .SelCount - 1
      System.Diagnostics.Debug.WriteLine(.get_SelectDate(i).ToString())
   Next
End With
```

The following C# sample prints the selected date(s):

```
for (int i = 0; i < axCalendar1.SelCount; i++)
   System.Diagnostics.Debug.WriteLine(axCalendar1.get_SelectDate(0).ToString());
```

The following VFP sample prints the selected date(s):

```
with thisform.Calendar1
   local i, d
   for i = 0 to .SelCount -1
      d = .SelectDate(i)
   next
endwith
```

property Calendar.Template as String

Specifies the control's template.

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

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

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 Calendar.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 / TemplatePut method 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.

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, 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.*

method Calendar.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 Calendar.TodayCaption as String

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

Type	Description
String	A string expression that indicates the caption for today button.

Use the TodayCaption property to specify the caption for combo's today button. Use the ShowTodayButton property to hide the control's today button. By default the TodayCaption property is "Today" .

property Calendar.ToolTipDelay as Long

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

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

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

property Calendar.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

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

property Calendar.ToolTipPopDelay as Long

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

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

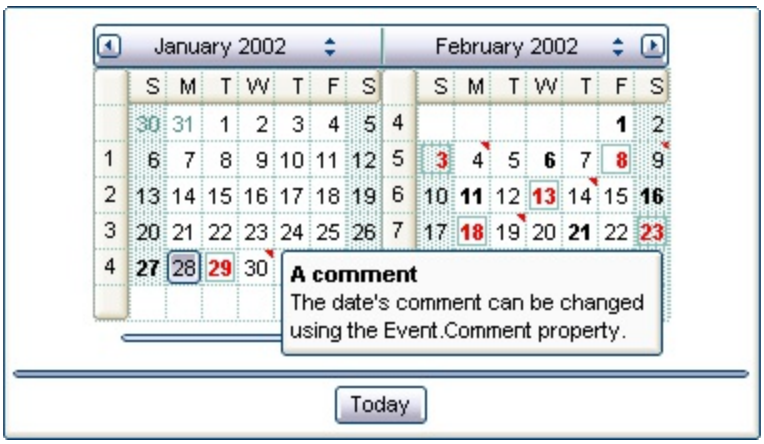
If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to assign a font for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

property Calendar.ToolTipWidth as Long

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

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

Use the ToolTipWidth property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipFont property to assign a font for the control's tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip.



method Calendar.UnSelDate (d as Date)

Unselects the date.

Type	Description
d as Date	A DATE expression that is going to be unselected.

Use the UnSelDate property to unselect a date. Use the SingleSel property to specify whether the control supports single or multiple selection. Use the FocusDate property to specify the date that has the focus. Use the SelectDate property to retrieve the selected date. Use the SelCount property to retrieve the number of selected dates.

The following VB sample unselects all dates:

```
With Calendar1
   While .SelCount() > 0
      .UnSelDate .SelectDate(0)
   Wend
End With
```

The following C++ sample unselects all dates:

```
while ( m_calendar.GetSelCount() > 0 )
   m_calendar.UnSelDate( m_calendar.GetSelectDate( 0 ) );
```

The following VB.NET sample unselects all dates:

```
With AxCalendar1
   While .SelCount > 0
      .UnSelDate(.get_SelectDate(0))
   End While
End With
```

The following C# sample unselects all dates:

```
while (axCalendar1.SelCount > 0)
   axCalendar1.UnSelDate(axCalendar1.get_SelectDate(0));
```

The following VFP sample unselects all dates:

```
with thisform.Calendar1
```

```
    do while ( .SelCount > 0 )
       .UnSelDate( .SelectDate(0) )
    enddo
endwith
```

property Calendar.UseVisualTheme as UIVisualThemeEnum

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

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

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

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

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

property Calendar.Value as Date

Specifies the selected date.

Type	Description
Date	A Date expression that indicates the selected date.

Use the Value property to access the selected date.

property Calendar.Version as String

Retrieves the control's version.

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

The Version property identifies the control's version.

property Calendar.VisualAppearance as Appearance

Retrieves the control's appearance.

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

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

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

- months, weeks, days header, Background property, HeaderBackColor property,
- up down, left or right arrows, Background property
- selected date(s), SelBackColor property
- events, BackColor property
- Today button, scrolling dates area, Background property
- today date, MarkToday property
- selected items in the months selector, Background property

property Calendar.VisualDesign as String

Invokes the control's VisualAppearance designer.

Type	Description
String	A String expression that encodes the control's Visual Appearance.

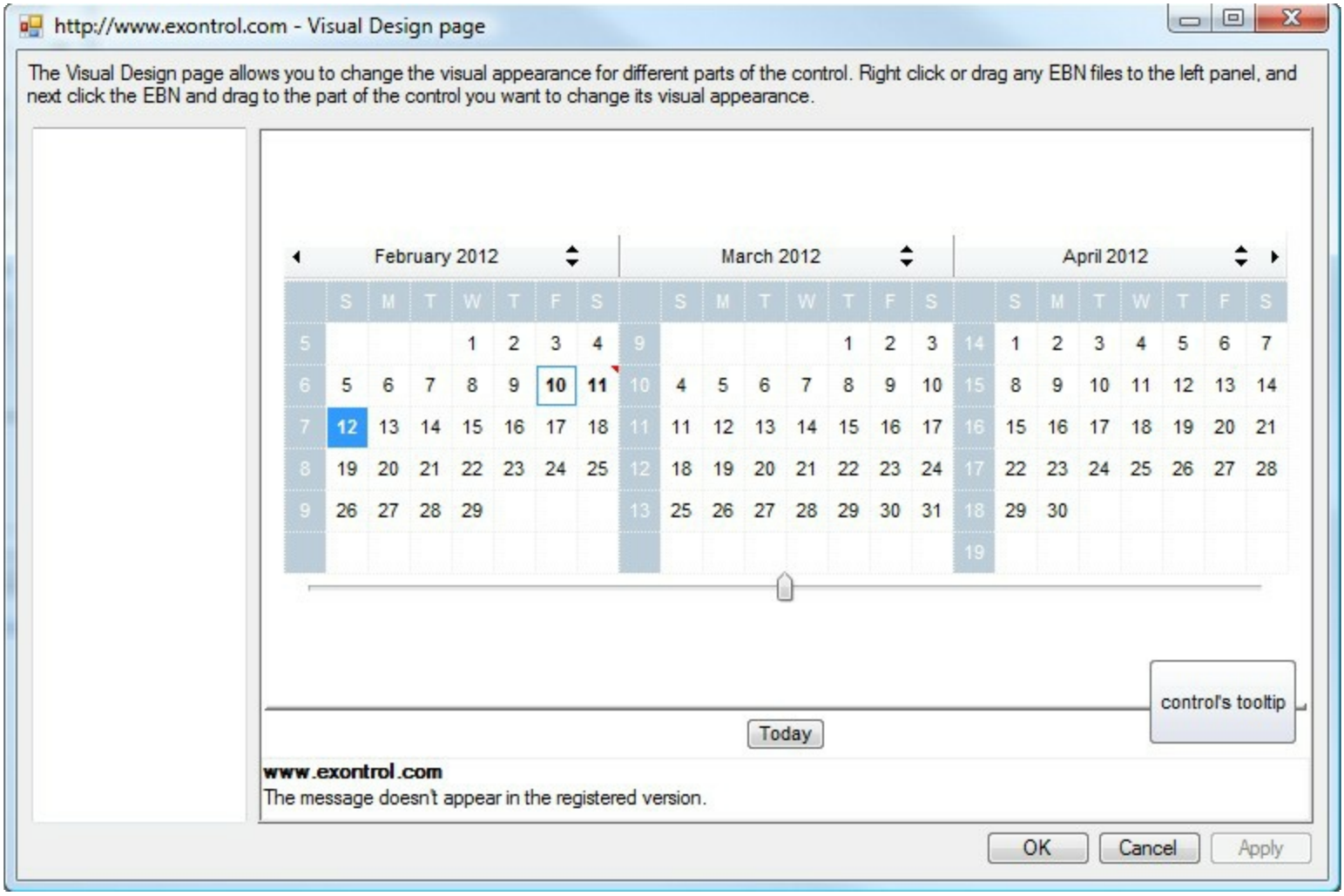
By default, the VisualDesign property is "". The VisualDesign property helps you to define fast and easy the control's visual appearance using the XP-Theme elements or EBN objects. The VisualDesign property can be accessed on design mode, and it can be used to design the visual appearance of different parts of the control by drag and drop XP or EBN elements. The VisualAppearance designer returns an encoded string that can be used to define different looks, just by calling the VisualDesign = encoded_string. If you require removing the current visual appearance, you can call the VisualDesign on "" (empty string). The VisualDesign property encodes EBN or XP-Theme nodes, using the Add method of the Appearance collection being accessed through the VisualAppearance property.

- For the /COM version, click the control in Design mode, select the Properties, and choose the "Visual Design" page.
- For the /NET version, select the VisualDesign property in the Properties browser, and then click ... so the "Visual Design" page is displayed.

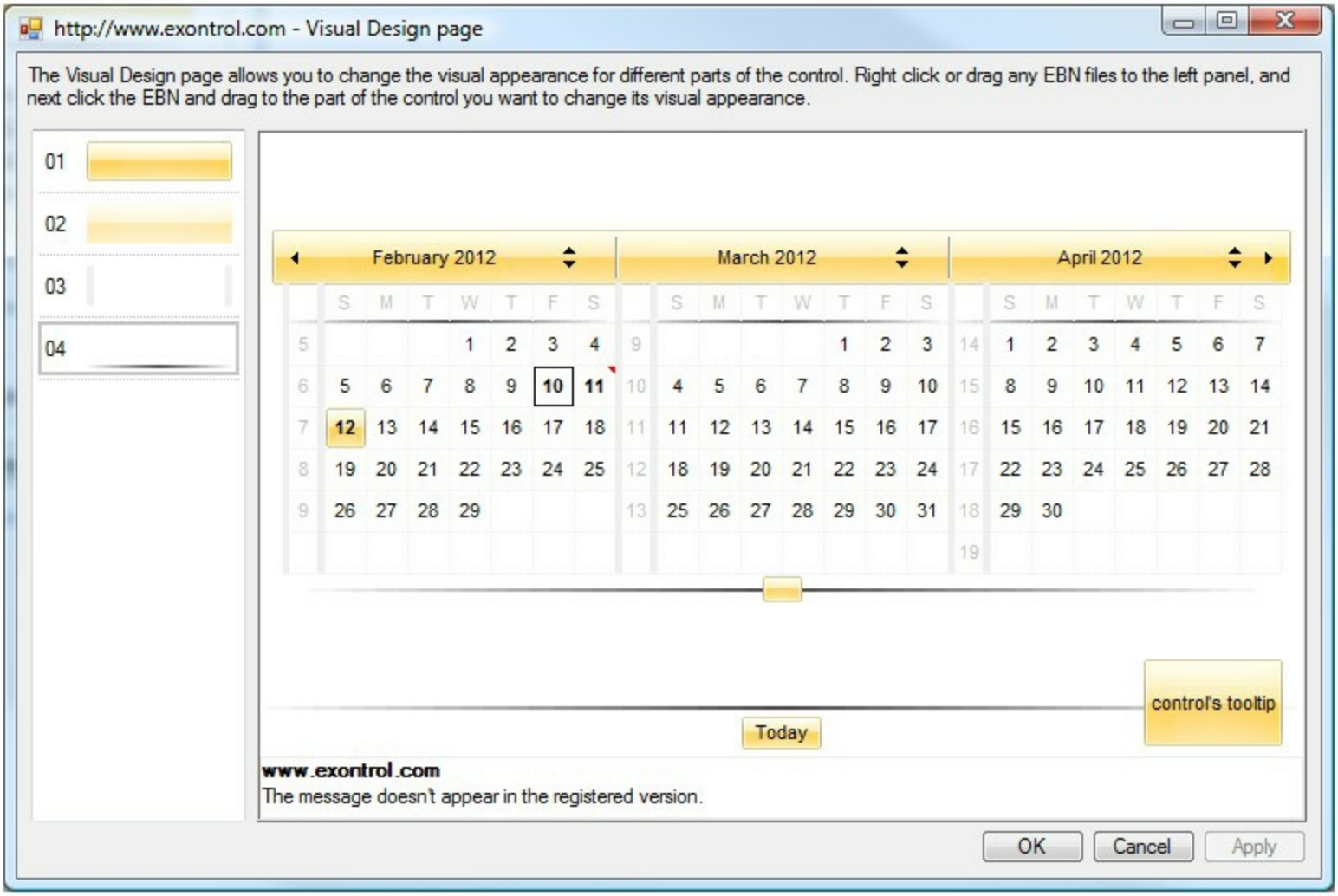
- Click here to watch a movie on how easily you can define the control's visual appearance using the XP-Theme or EBN files.

The left panel, should be user to add your EBN or XP-Theme elements. Once you add them drag and drop the EBN or XP-Theme element from the left side to the part which visual appearance you want to change.

The following picture shows the control's VisualDesign form (empty):



The following picture shows the control's VisualDesign form after applying some EBN objects:



This layout generates the following code:

```
With Calendar1
   .VisualDesign =
"gBFLBWIgBAEHhEJAEGg6TC8HHxoBYABkIdT/jDAjQAdcYf8aYAAfkehIDhQAAQEk4BA0risKA
 & _
"hehaJgJHYYYXmYKYWGSGZmDmJhmhqJoJi4bobGcCYyHKG5nDmRh0h2ZxpmYdYemeCZy
 & _
"I+B/A3EiOUflXg3iTHOiURwyx0jtF8KYMwRQzj+gwO0PYnhni9HaPsfw0QFifH5JEZY8x+CfCe
 & _
"j0HgJ8CgDg+hQCyPMII6ACD9FCFMfwhRoCaAyEMcgfBGCiCoEYRQSX9AjE0BVMo9AzCO
 & _
"CMAgYwHgrAKI0IYbggDICwP8EQZAOCL/II7PAQx/gkHIMwS4kRZCGCGMoHIHwVAyBuBWI
 & _
"ZA6BPBehpAdIEh2AugDhZA5gnA+hlg8gvg0h2gOhXhZh7A4ByhGg9x7h3GEBcA7gvByhuh
 & _
"BHgRgBAIATgHgoB6gAgoAKgHgHgQgHhIASgIAmhngHhoANgIBoAOB9gngEgIg3gDgIhX
 & _
```

```
"pgIgYg5gKgYhZgJgYh5AOgZAZQUA4h1AKBohyAKXIgKAYhyAJ4bAOhjBAgOhAB5AKgghI
 & _
"gpgQ7bgghxgph66SgK6ehagkgqB6gj5FGhAqgogqg6gugqhagwgqhKgygqh6gsgoAQhC
 & _
"7gbgtg6g7g1wXA2g7hrhAg7gLg7g8A7hCg6h7g/g7hbg9ANASgIg8zoA8hAhSAogLhPAP
 & _
"GE7x5Lk4gZPmGi5OMiSKIUiRBBYFzXJQHzBP4Gy0LcoAZP8oA6AkoB6A0oBaAQGzULcoW
 & _
"ScF0xR8EsxQcGMxh8E8xC8F/dBhMftTGFwazHLweTFDwczGPwbgdJgTYyEyDIGCNsRIvBJAfI
 & _
"BuBiIYAABgRghGICYFgABCBFHGFASIIA5ATAyIwIw4hogYEQMQNAMAAhCCMIMLIABpBTAG
 & _
"IoKA7AegPGIBEBgPAShEEiCwJoExigRCYFUCQxAohsC6A0YwERGAtASMYSIrAMgbGOBAHg

End With
```

If running the empty control we get the following picture:

If running the control using the code being generated by the VisualAppearance designer we get:

property Calendar.WeekDayName(WeekDay as WeekDayEnum) as String

Retrieves or sets a value that indicates the week day short name in the week days header.

Type	Description
WeekDay as WeekDayEnum	A WeekDayEnum expression that indicates the day of the week.
String	A String expression that specifies the name of the day in the week.

Use the WeekDayName property to change the name of a specified day of the week. Use the WeekDays property to assign a name for all days in the week.

property Calendar.WeekDays as String

Retrieves or sets a value that indicates the list of short names for each week day, separated by space.

Type	Description
String	A string expression that indicates the list of short names for each week day, separated by space.

By default, the WeekNames is "S M T W T F S". Use the FirstDay property to change the first day of the week. Use the MonthNames property to define the name for each month. The LocWeekDays property specifies the name of the days in the week, using the current user regional and language settings.

The following sample shows how to use three letters for each week day:

```
Private Sub Form_Load()
   With Calendar1
      .WeekDays = "Sun Mon Tue Wed Thu Fri Sat"
      .AutoSize = False
      .FixedCellHeight = 18
      .FixedCellWidth = 32
   End With
End Sub
```

CalendarCombo 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: {1229B856-7540-4AF7-A53D-53B00FB8CF6B}. The object's program identifier is: "ExCalendar.CalendarCombo". The /COM object module is: "ExCalendar.dll"

The CalendarCombo is the drop-down version of the Calendar object. The CalendarCombo object supports the following properties and methods:

Name	Description
Alignment	Specifies the alignment for the drop down portion of the control.
AlignmentDay	Specifies the alignment of the days within the control.
AllowCheckBox	Specifies whether the label displays a checkbox to the left of the date. When unchecked, no date is selected.
AllowEditChanges	Specifies whether the control's edit box allows changes.
AllowSpin	Specifies whether the control's label displays a spin to change the date.
AMPM	Specifies the AM and PM indicators.
Appearance	Retrieves or sets the control's appearance
AppearanceDay	Retrieves or sets a value that determines the day's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoAdvance	Specifies whether the next field is focused, once that the user fills data in the focused field.
AutoSize	Retrieves or sets a value that indicates whether the control automatically resizes the cell based on the size of the font.
BackColor	Retrieves or sets the control's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BorderLineColor	Retrieves or sets a value that indicates the border line color.
Caption	Specifies the caption/date's element being displayed in the control's label.
CheckImage	Retrieves or sets a value that indicates the image being displayed as a checkbox.

CommentBackColor	Retrieves or sets the color to highlight the commented events.
Date	Retrieves or sets the browsed date. Ensures that the date is visible.
DateHeaderField	Specifies the HTML caption to be shown on the giving field of the date's header.
DateHeaderFormat	Specifies the CRD format to display the month/year/buttons within the date's header.
Debug	Gets debugging information.
DoDate	Composes a DATE type, based on year, month and day.
DrawBorderLine	Retrieves or sets a value that indicates the border line style.
DrawGridLine	Retrieves or sets a value that identifies the type of grid lines.
DropDown	Specifies whether the drop down is visible or hidden.
Enabled	Retrieves or sets a value that indicates whether the control is enabled ot disabled.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
Events	Retrieves the control date events collection.
ExecuteTemplate	Executes a template and returns the result.
FirstDay	Retrieves or sets a value that indicates the first day of the week.
FirstVisibleDate	Retrieves the first visible date.
FixedCellHeight	Retrieves or sets a value that indicates the cell's height while the AutoSize is exFixed.
FixedCellWidth	Retrieves or sets a value that indicates the cell's width while the AutoSize is exFixed.
FocusIndex	Specifies the index of the focused element.
Font	Retrieves or sets the font to display the drop down portion of the control.
ForeColor	Retrieves or sets the control's foreground color.
ForeColorDisabled	Retrieves or sets the control's foreground color when the date is locked.
	Formats the A,B,C values based on the giving expression

FormatABC	and returns the result.
FormatDate	Retrieves or sets a value that indicates the format of date displayed.
FormatUserDate	Retrieves or sets a value that indicates the string format of the date while FormatDate is UserDate.
FreezeEvents	Prevents the control to fire any event.
GridLineColor	Retrieves or sets a value that indicates the grid lines color.
HeaderBackColor	Retrieves or sets a value that indicates the background color used for weeks and week days headers.
HeaderForeColor	Retrieves or sets a value that indicates the foreground color used for weeks and week days headers.
HideDropDownButton	Determines whether the drop down button is visible or hidden when the control loses the focus.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets the control's handle image list.
ImageSize	Retrieves or sets the size of icons the control displays..
IndexFromPoint	Retrieves the index of the date's element from the point.
LabelBounds	Specifies the bounds to display the control's label.
LabelFont	Retrieves or sets the label's font.
LabelHeight	Specifies the label's height.
LastVisibleDate	Retrieves the last visible date.
Locked	Specifies whether the user can change the selection.
MarkToday	Retrieves or sets a value that indicates whether the control marks the today date.
MaskOnEmpty	Specifies the masking string for each entity when the date is empty.
MaxDate	Retrieves or sets the min date.
MaxMonthX	Specifies the maximum number of months horizontally displayed.
MaxMonthY	Specifies the maximum number of months vertically displayed.
MaxScrollYear	Specifies the maximum year when scrolling.

MinDate	Retrieves or sets the min date.
MinMonthX	Specifies the minimum number of months horizontally displayed.
MinMonthY	Specifies the minimum number of months vertically displayed.
MinScrollYear	Specifies the minimum year when scrolling.
MonthName	Retrieves or sets the month's name.
MonthNames	Retrieves or sets a value that indicates the list of month names, separated by space.
NonMonthDaysColor	Retrieves or sets a value that indicates the color to show the non-month days.
NonworkingDays	Retrieves or sets a value that indicates the non-working days, for each week day a bit.
NonworkingDaysColor	Retrieves or sets a value that indicates the color to fill the non-working days.
NonworkingDaysForeColor	Retrieves or sets a value that indicates the foreground color for non-working days.
NonworkingDaysPattern	Retrieves or sets a value that indicates the pattern being used to fill non-working days.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
ScrollOnDrop	Specifies a value that indicates whether the drop down calendar is scrolled when it is shown.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
SelDate	Selects a date.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowDays	Retrieves or sets a value that indicates whether the week days header is visible or hidden.
ShowFocusRect	Specifies whether the focus rectangle is shown around the label while the control has the focus.
ShowImageList	Specifies whether the control's image list window is visible or hidden.

ShowMonth	Retrieves or sets a value that indicates whether the month header is visible or hidden.
ShowMonthSelector	Retrieves or sets a value that indicates whether the user is able to select a new month by clicking in the month header.
ShowNonMonthDays	Specifies whether the control displays the dates that are not part of the month.
ShowTodayButton	Retrieves or sets a value that indicates whether the today button is visible or hidden.
ShowWeeks	Retrieves or sets a value that indicates whether the weeks header is visible or hidden.
ShowYearScroll	Retrieves or sets a value that indicates whether the scroll bar for changing the year is visible or hidden.
ShowYearSelector	Retrieves or sets a value that indicates whether the year selector is visible or hidden.
Template	Specifies the control's template.
TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.
TodayCaption	Retrieves or sets a value that indicates the today button's caption.
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.
UseVisualTheme	Specifies whether the control uses the current visual theme to display certain UI parts.
Value	Retrieves or sets the browsed date. Ensures that the date is visible.
VisualAppearance	Retrieves the control's appearance.
WaitAutoAdvance	Specifies the time in ms to wait until the selection moves to the next editing field in the CalendarCombo control.
WeekDayName	Retrieves or sets a value that indicates the week day short name in the week days header.

WeekDays	Retrieves or sets a value that indicates the list of short names for each week day, separated by space.

property CalendarCombo.Alignment as AlignmentEnum

Specifies the alignment for the drop down portion of the control.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the drop down portion of the control relative to the control's label. If the Alignment property is 16, the width of the drop down portion is the same as the width of the control's label, in other words the drop down portion is aligned to the control's label.

By default, the Alignment property is RightAlignment, which makes the rightmost portion of the control being aligned with the rightmost portion of the control's label. The AlignmentDay property specifies the day's alignment relative to its cell. Use the Alignment property to specify a different alignment for the drop down portion of the control. The size of the drop down portion of the control is automatically computed based on the control's AutoSize, Font, MaxMonthX and MaxMonthY properties.

property CalendarCombo.AlignmentDay as AlignmentEnum

Specifies the alignment of the days within the control.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the day within the cell.

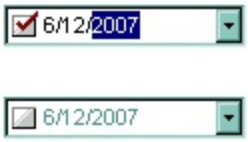
By default, the AlignmentDay property is RightAlignment, which means that the day is displayed to the right of their cells. Use the AlignmentDay property to specify the horizontal alignment of the day inside the cell where it is displayed. For instance, if the cell's size is fixed using the AutoSize property on exFixedSize, FixedCellWidth and FixedCellHeight property you can specify the day's alignment to be on center.

property CalendarCombo.AllowCheckBox as VisibleEnum

Specifies whether the label displays a checkbox to the left of the date. When unchecked, no date is selected.

Type	Description
VisibleEnum	A VisibleEnum expression that specifies whether the label displays a check box.

By default, the AllowCheckBox property is exHidden. Use the AllowCheckBox property to display a check box left to the date, so a date can be selected or not. Use the CheckImage property to use icons, or EBN files to specify your visual appearance for the control's checkbox. Use the AllowSpin property to display a spin control to change the focused field. Use the Value property to change the checkbox's state as follows: If the Value property is not zero, the check box is checked, else the checkbox is unchecked, and the labels shows grayed. Use the ForeColorDisabled property to specify the color to show the selected date when the checkbox is unchecked. For instance, change the ForeColorDisabled to be the control's BackColor and so no date is displayed when the Value property is 0 or the control's checkbox is un-checked. In VB/NET or C# you may need to use the Date.FromOADate(0) to convert the 0 value to a null date.



property CalendarCombo.AllowEditChanges as Boolean

Specifies whether the control's edit box allows changes.

Type	Description
Boolean	A boolean expression that specifies whether the control's edit box allows changes.

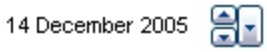
Use the AllowEditChanges property to specify whether the control's edit box allows changes using the keyboard. Use the AllowSpin property to show a spin in the control's label. Use the FormatDate property to specify the format of date being displayed in the control's label. Use the AutoAdvance property to disable moving the focused field while typing digits.

property CalendarCombo.AllowSpin as VisibleEnum

Specifies whether the control's label displays a spin to change the date.

Type	Description
VisibleEnum	A VisibleEnum expression that indicates whether the label of the control displays a spin to change the date.

By default, the AllowSpin property is exHidden. Use the AllowSpin property to display a spin in the control's label. Use the AllowEditChanges property to specify whether the control's edit box allows changes. The spin changes the field that has the focus. The field that displays the day gets the focus, if no field is highlighted, in the control's label. Use the FormatDate property to specify the format of date being displayed in the control's label. Use the Background property to change the visual appearance for the up and down arrows of the spin. Use the AutoAdvance property to disable moving the focused field while typing digits. Use the AllowCheckBox property to display a check box left to the date.



property CalendarCombo.AMPM as String

Specifies the AM and PM indicators.

Type	Description
String	A String expression that indicates the AM PM indicators, separated by space.

By default, the AMPM property is "AM PM". Use the AMPM property to change the AM/PM indicators when the control displays time, and the %a field is included in the FormatUserDate property. If the AMPM property is empty string, and the %a field is present, the hours will still be displayed in the AM/PM format, but the AM/PM indicators will not be displayed. if the %a field is not present, the hours get displayed in 24 hours format. The [Calendar.]LocAMPM property specifies specifies the AM and PM indicators, as indicated in the regional settings.

property CalendarCombo.Appearance as AppearanceEnum

Retrieves or sets the control's appearance

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

Defines how the control's border looks like. Use the Appearance property to remove the control's default border. Use the Add method to add new skins to the control. Use the BackColor property to specify the control's background color. Use the Background(exDropDownBackColor) property to specify the background color of the drop down portion of the control. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exDropDownAppearance) property to change the visual appearance of the drop down portion of the control.

The following VB sample changes the visual aspect of the borders of the control's label (please check the above picture for round corners):

```
With CalendarCombo1
   .BeginUpdate
      .VisualAppearance.Add &H16, "c:\temp\frame.ebn"
      .Appearance = &H16000000
      .BackColor = RGB(250, 250, 250)
   .EndUpdate
End With
```

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

```
With AxCalendarCombo1
   .BeginUpdate()
   .VisualAppearance.Add(&H16, "c:\temp\frame.ebn")
   .Appearance = &H16000000
   .BackColor = Color.FromArgb(250, 250, 250)
   .EndUpdate()
End With
```

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

```
axCalendarCombo1.BeginUpdate();
axCalendarCombo1.VisualAppearance.Add(0x16, "c:\\temp\\frame.ebn");
axCalendarCombo1.Appearance =
(EXCALENDARCOMBOLib.AppearanceEnum)0x16000000;
axCalendarCombo1.BackColor = Color.FromArgb(250, 250, 250);
axCalendarCombo1.EndUpdate();
```

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

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

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

```
with thisform.CalendarCombo1
    .BeginUpdate
        .VisualAppearance.Add(0x16, "c:\temp\frame.ebn")
        .Appearance = 0x16000000
        .BackColor = RGB(250, 250, 250)
    .EndUpdate
endwith
```

property CalendarCombo.AppearanceDay as AppearanceDayEnum

Retrieves or sets a value that determines the day's appearance.

Type	Description
AppearanceDayEnum	An AppearanceDayEnum expression that defines the date's appearance.

Defines the way how border's date looks like. The control defines two types of date's appearance: flat and 3D.

The following screen shot shows the control using the DayFlat value:

The following screen shot shows the control using the Day3D value:

method CalendarCombo.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 CalendarCombo1_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 CalendarCombo.AutoAdvance as AutoAdvanceEnum

Specifies whether the next field is focused, once that the user fills data in the focused field.

Type	Description
AutoAdvanceEnum	An AutoAdvanceEnum expression that indicates whether the next field is focused, once that the user fills data in the focused field.

By default, the AutoAdvance property is exAdvanceCycle. Use the AutoAdvance property to disable moving the focused field, while user type data. Use the AllowEditChanges property to disable entering new data using the keyboard. Use the AllowSpin property to assign spin to the control's label. Use the FormatUserDate property to specify the fields in the control's label. For instance, if the AutoAdvance property is True, and the user types 4 in the day field, the next field is automatically focused, because there is no day with two digits, and starts with 4. If the user types 1, the next field is focused only after a certain time is elapsed. The same idea is for the month field. If the user types 4, the next field is automatically focused, because there is no month with two digits and starts with 4. Use the WaitAutoAdvance property to specify the time in ms to wait until the selection moves to the next editing field in the CalendarCombo control.

property CalendarCombo.AutoSize as AutoSizeEnum

Retrieves or sets a value that indicates whether the control automatically resizes the cell based on the size of the font.

Type	Description
AutoSizeEnum	An AutoSizeEnum expression that indicates whether the control automatically resizes the cell based on the size of the font.

By default, the AutoSize property is exFontSize. The Font property specifies the control's font. The MaxMonthX property specifies the number of months that can displayed on the horizontal axis. The MaxMonthY property specifies the number of months that can displayed on the vertical axis. Use the AutoSize, FixedCellHeight and FixedCellWidth properties to defines the size of the control's cell. A cell displays a date. The FixedCellHeight and FixedCellWidth properties has effect only if the AutoSize is False.

The following sample fixes the cell's size to (18,32):

```
Private Sub Form_Load()
   With CalendarCombo1
      .AutoSize = exFixedSize
      .FixedCellHeight = 18
      .FixedCellWidth = 32
   End With
End Sub
```

property CalendarCombo.BackColor as Color

Retrieves or sets the control's background color.

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

Use the ForeColor property to change the foreground color. Use the Background(exDropDownBackColor) property to specify the background color of the drop down portion of the control. Use the BackColor property of Event object to change the cell's background color. Use the Background(exDropDownAppearance) property to change the visual appearance of the drop down portion of the control.

property CalendarCombo.Background(Part as BackgroundPartEnum) as Color

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

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

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

The following VB sample changes the visual appearance for the drop down button. The sample uses the "" skin when the drop down button is up, and the "" when it is down.

```
With CalendarCombo1
  With .VisualAppearance
    .Add &H25, "D:\Temp\ExCalendar.Help\dropup.ebn"
    .Add &H26, "D:\Temp\ExCalendar.Help\dropdown.ebn"
  End With
  .Background(exDropDownButtonUp) = &H25000000
```

```
    .Background(exDropDownButtonDown) = &H26000000
End With
```

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

```
m_calendarcombo.GetVisualAppearance().Add( 0x25,
COleVariant("D:\\Temp\\ExCalendar.Help\\dropup.ebn"));
m_calendarcombo.GetVisualAppearance().Add( 0x26,
COleVariant("D:\\Temp\\ExCalendar.Help\\dropdown.ebn"));
m_calendarcombo.SetBackground( 4 /*exDropDownButtonUp*/, 0x25000000 );
m_calendarcombo.SetBackground( 5 /*exDropDownButtonDown*/, 0x26000000 );
```

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

```
With AxCalendarCombo1
    With .VisualAppearance
        .Add(&H25, "D:\Temp\ExCalendar.Help\dropup.ebn")
        .Add(&H26, "D:\Temp\ExCalendar.Help\dropdown.ebn")
    End With
    .set_Background(EXCALENDARLib.BackgroundPartEnum.exDropDownButtonUp,
&H25000000)
    .set_Background(EXCALENDARLib.BackgroundPartEnum.exDropDownButtonDown,
&H26000000)
End With
```

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

```
axCalendarCombo1.VisualAppearance.Add(0x25,
"D:\\Temp\\ExCalendar.Help\\dropup.ebn");
axCalendarCombo1.VisualAppearance.Add(0x26,
"D:\\Temp\\ExCalendar.Help\\dropdown.ebn");
axCalendarCombo1.set_Background(EXCALENDARLib.BackgroundPartEnum.exDropDownBu
 0x25000000);
axCalendarCombo1.set_Background(EXCALENDARLib.BackgroundPartEnum.exDropDownBu
 0x26000000);
```

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

```
With thisform.CalendarCombo1
```

```
    With .VisualAppearance
        .Add(37, "D:\Temp\ExCalendar.Help\dropup.ebn")
        .Add(38, "D:\Temp\ExCalendar.Help\dropdown.ebn")
    EndWith
    .Background(4) = 620756992 && exDropDownButtonUp
    .Background(5) = 637534208 && exDropDownButtonDown
EndWith
```

where the 620756992 is 0x25000000 in hexa, and 37 is 0x25 in hexa

property CalendarCombo.BorderLineColor as Color

Retrieves or sets a value that indicates the border line color.

Type	Description
Color	A color expression that indicates the month's border line color.

Use the Appearance property to change the control's border. Use the DrawBorderLine property to define the style of month's border line, or to hide it.

property CalendarCombo.Caption ([Index as Variant]) as String

Specifies the caption being displayed in the control's label.

Type	Description
Index as Variant	A long expression that indicates the index of the element being retrieved. If missing, the entire caption is retrieved.
String	A String expression that indicates the caption being displayed in the control's label.

Use the Caption property to retrieve the caption being displayed in the control's label. Use the SelDate property to get or set the date being displayed in the control. Use the FocusIndex property to specify the index of the element being focused in the control's label. The SelectionChanged event is fired when the user changes the selected date. Use the AutoAdvance property to automatically move the focused element to the next element. Use the FormatDate property to specify the format of the date being displayed.

property CalendarCombo.CheckImage(State as CheckStateEnum) as Long

Retrieves or sets a value that indicates the image being displayed as a checkbox.

Type	Description
State as CheckStateEnum	A CheckStateEnum expression that specifies the state of the checkbox whose image is changed
Long	A Long expression that indicates the index of the icon being displayed, in the specified state.

Use the CheckImage property to change the visual appearance for the control's check box. The ImageSize property defines the size (width/height) of the check-box field. Use the AllowCheckBox property to display a checkbox left to the control's label. Use the Images method to assign a list of icons to the control.

property CalendarCombo.CommentBackColor as Color

Retrieves or sets the color to highlight the commented events.

Type	Description
Color	A Color expression that specifies the color to mark the dates that have a comment or a tooltip assigned, 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 for dates with comments assigned

The CommentBackColor property indicates the color to mark the dates with the Comment property being set. If the CommentBackColor property is identical with the BackColor property, the dates are not marked as they have assigned a tooltip or a comment.

property CalendarCombo.Date as Date

Retrieves or sets the browsed date. Ensures that the date is visible.

Type	Description
Date	A DATE expression that indicates the browsed date.

Use the Date property to browse for a date, in the drop down portion of the control. The SelDate property indicates the selected date. The Value property is identical with the SelDate property. The DateChanged event is fired when the user changes the browsed date. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date.

property CalendarCombo.DateHeaderField(Field as HeaderFieldEnum) as String

Specifies the HTML caption to be shown on the giving field of the date's header.

Type	Description
Field as HeaderFieldEnum	A HeaderFieldEnum expression that defines the index of the part to be changed.
String	A string expression that specifies the HTML caption to be shown on the giving field of the date's header. The DateHeaderField property could be also an expression whose result, can be the HTML caption to be shown on the giving field of the date's header. The **<%month%>** specifies the name of the browsed month, while the **<%year%>** defines the year of the browsed month

By default, the DateHeaderField property for:

- exHeaderDate is "<c><%month%> <%year%>"
- exHeaderPrevMonth is "<c><sha ;;0>-"
- exHeaderNextMonth is "<c><sha ;;0>+"
- exHeaderPrevYear is "<c><sha ;;0><"
- exHeaderNextYear is "<c><sha ;;0>>"

The DateHeaderFormat property specifies the CRD format to display the month/year/buttons within the date's header. The DateHeaderField property has effect only if the DateHeaderFormat property is not empty. The DateHeaderField property could be also an expression that returns the HTML caption to be displayed on the giving field of the date's header. In other words, if the expression of the DateHeaderField property is not valid, it indicates directly the HTML caption, else the HTML caption is the result of evaluating the giving expression.

For instance:

- "<c><%month%> <%year%>", shows the month and the year centered
- "<c><img>up</img>", displays the image named up centered. Previously, the HTMLPicture("up") should be called to specify the location of the picture named up.
- "`<c><%month%> (` + (dateF(value) left 2) + `) <r><off -6><font ;6><%year%>`" includes the month number in the date's header
- " (month(value) = month(date(``)) ? `<b><fgcolor=0000FF>` : ``) + `<c><%month%> <%year%>`" shows the current month with a different foreground color (bold, blue)

The DateHeaderField property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.
- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect.

By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The 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 expression of the DateHeaderField property could use keywords such as:

- **value** which indicates the date of the month to be formatted

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 CalendarCombo.DateHeaderFormat as String

Specifies the CRD format to display the month/year/buttons within the date's header.

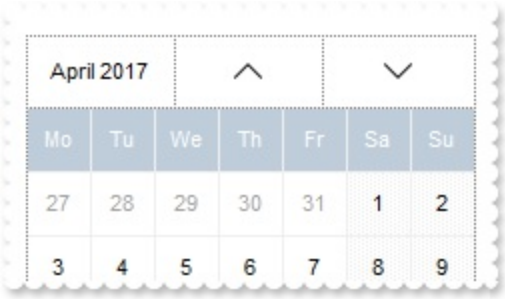
Type	Description
String	A String expression that defines the CRD format to display the month/year/buttons within the date's header. The value of the DateHeaderFormat property could be also an expression that returns a CRD format. You can use the eXCRD tool to define, edit and view CRD strings.

By default, the DateHeaderFormat property is "", which indicates that it has no effect. The DateHeaderFormat property helps you to customize the date's header which includes the month, year and the prev / next month/year buttons. The DateHeaderFormat property could be also an expression that returns a CRD format for specified month. In other words, if the expression of the DateHeaderFormat property is not valid, it indicates directly the CRD syntax, else the CRD syntax is the result of evaluating the giving expression. The DateHeaderField property specifies the HTML caption to be shown on the giving field of the date's header.

The following screen show shows the control with a different date header:

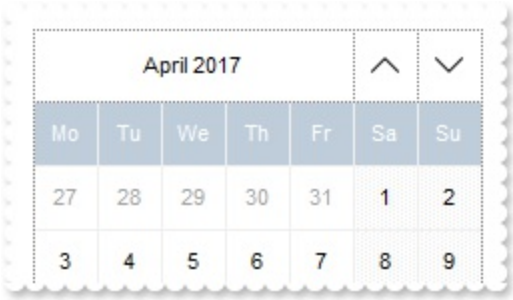
For instance, here are few simple CRD strings:

- The CRD string "`**1,2,3**`" divides the header in three parts, the left side displays the month/year, the middle part displays the next month button, while the last part displays the next month button. Similar with horizontally splitting a cell in three pieces.



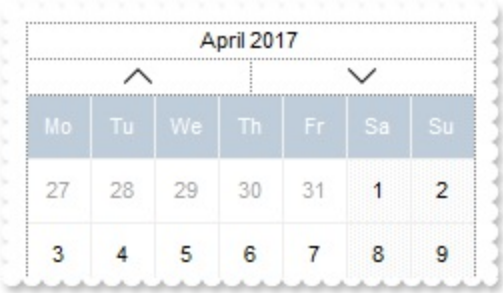
- The CRD string "`**1,2:32,3:32**`" divides the header in three parts, the left side displays the month/year, the middle part displays the next month button, while the last part

displays the next month button. The last two-parts of the CRD are 32-pixels wide.



- The CRD string "`**1,(2/3):32**`" splits horizontally the header in two parts, where the left part displays the month/year while the right-part of 32-pixels wide is divided in two parts, the upper part displays the prev month button and the bottom part displays the next month button.

- The CRD string "`**1/2,3**`" splits the header in two, the upper part displays the month/year, the bottom part is divided in other two parts, where the left part displays the prev month button, and the right part displays the next month button.



You can use the eXCRD tool to define, edit and view CRD strings.

The DateHeaderFormat property can include any of the following:

- **1** index in the CRD format represents the month/year. The DateHeaderField(exHeaderDate) / DateHeaderField(1) property defines what an 1-index part of the CRD string displays. By default, it displays the month and the year of the browsed date. Clicking the 1-index part shows the control's months selector, so the user can select a different month/year.
- **2** specifies the button to go previously one month. The DateHeaderField(exHeaderPrevMonth) / DateHeaderField(2) property defines what an 2-index part of the CRD string displays. By default, it displays "-" character. Clicking

the 2-index part navigates the control to previously month.
- **3** specifies the button to advance to the next month. The DateHeaderField(exHeaderNextMonth) / DateHeaderField(3) property defines what an 3-index part of the CRD string displays. By default, it displays "+" character. Clicking the 3-index part navigates the control to next month.
- 4 specifies the button to go previously one year. The DateHeaderField(exHeaderPrevYear) / DateHeaderField(4) property defines what an 4-index part of the CRD string displays. By default, it displays "<" character. Clicking the 4-index part navigates the control to previously year.
- **5** specifies the button to advance one year. The DateHeaderField(exHeaderNextYear) / DateHeaderField(5) property defines what an 5-index part of the CRD string displays. By default, it displays ">" character. Clicking the 5-index part navigates the control to next year.

For instance, DateHeaderFormat property on

- "1,2:24,3:24" displays the month/year(1), and aligned to the right with a 24-pixels wide the prev(2) and next(3) month-buttons.
- "month(value) = 1 ? `4:24,5:24,1,2:24,3:24` : `1,2:24,3:24`", specifies for January month to include all buttons, and for the rest just the prev and next month-buttons.

The expression of the DateHeaderFormat property could use keywords such as:

- **value** which indicates the date of the month to be formatted
- **x** indicates the x-position of the month within the calendar
- **xmax** specifies the number of months being displayed horizontally in the calendar
- **y** indicates the y-position of the month within the calendar
- **ymax** specifies the number of months being displayed vertically in the calendar.

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 CalendarCombo.Debug ([Options as Variant]) as String

Gets debugging information.

Type	Description
Options as Variant	Reserved.
String	Reserved.

For internal use only.

property CalendarCombo.DoDate (Year as Long, Month as Long, Day as Long) as Date

Composes a DATE type, based on year, month and day.

Type	Description
Year as Long	A long expression that indicates the year.
Month as Long	A long expression that indicates the month. 1 - January, 2 - February, ... 12 - December
Day as Long	A long expression that indicates the day number. 1 - First day of the month, ... If the Day is -1, the DoDate returns the last day of the specified month/year.
Date	A DATE expression that indicates the built date

Use the DoDate to build a DATE expression given the year, month and the day. If the Day is -1, the DoDate returns the last day of the specified month/year.

property CalendarCombo.DrawBorderLine as LineStyleEnum

Retrieves or sets a value that indicates the border line style.

Type	Description
LineStyleEnum	A LineStyleEnum expression that indicates the month's border line style.

Use the BorderLineColor property to change the color for the month's border. Use the DrawBorderLine to hide the month's border line.

property CalendarCombo.DrawGridLine as LineStyleEnum

Retrieves or sets a value that identifies the type of grid lines.

Type	Description
LineStyleEnum	A LineStyleEnum expression that indicates whether the grid lines are visible or hidden.

Use the DrawGridLine property to show the grid lines. Use the GridLineColor property to define the color for grid lines.

property CalendarCombo.DropDown([Reserved as Variant]) as Boolean

Shows or hides programmatically the drop down portion of the control.

Type	Description
Reserved as Variant	Reserved value.
Boolean	A Boolean expression that specifies whether the drop down portion of the control is visible or hidden.

Use the DropDown method to shows or hides programmatically the drop down portion of the control. Use the Value property to change the control's date. Use the AllowSpin property to display a spin control inside the control's label. Use the Background(exDropDownButtonUp) property to change the visual appearance for the drop down button of the control. You can hide the control's drop down button using the HideDropDownButton property.

property CalendarCombo.Enabled as Boolean

Retrieves or sets a value that indicates whether the control is enabled ot disabled.

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

Use the Enabled property to disable the control.

property CalendarCombo.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 CalendarCombo.Events as Events

Retrieves the control date events collection.

Type	Description
Events	An Events collection that contains Event objects. Each Event defines a DATE and its graphical information like: foreground color, background color, font attributes, tooltips, images and so.

Use the Events property to access the Event objects. The following sample applies a bold effect to "yesterday" date:

```
CalendarCombo1.Events.Add(Date - 1).Bold = True
```

method CalendarCombo.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 number of events within the control:

```
Debug.Print CalendarCombo1.ExecuteTemplate("Events.Count")
```

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.

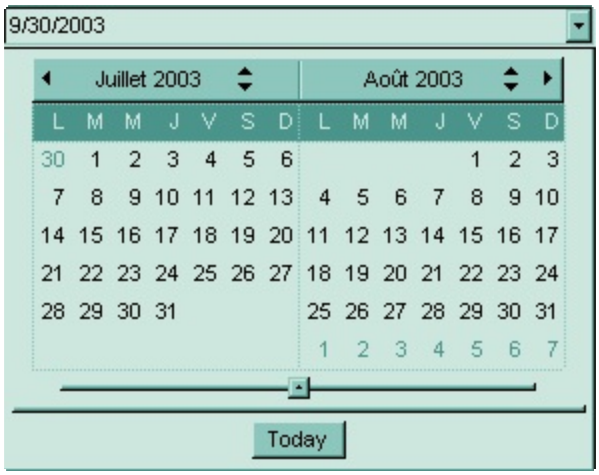
property CalendarCombo.FirstDay as WeekDayEnum

Retrieves or sets a value that indicates the first day of the week.

Type	Description
WeekDayEnum	A WeekDayEnum expression that defines the first day of the week.

Use the FirstDay property to define the month for a specific language. Use the WeekDays property to define the shortcut for each week day. Use the MonthNames to define the name for each month. For instance, in Europe, the week starts on Monday. The following sample defines the "french" style:

```
Private Sub Form_Load()
   With CalendarCombo1
      .FirstDay = Monday
      .MonthNames = "Janvier Février Mars Avril Mai Juin Juillet Août Septembre Octobre
Novembre Décembre"
      .WeekDays = "D L M M J V S"
   End With
End Sub
```



property CalendarCombo.FirstVisibleDate as Date

Retrieves the first visible date.

Type	Description
Date	A DATE expression that specified the first visible date in the calendar.

The FirstVisibleDate property retrieves the first visible date being displayed in the calendar. Use the ShowNonMonthDays property to display the dates that are not part of the month. Use the LastVisibleDate property to get the last visible date being displayed in the calendar.

property CalendarCombo.FixedCellHeight as Long

Retrieves or sets a value that indicates the cell's height while the AutoSize is false.

Type	Description
Long	A long expression that defines the cell's height while the AutoSize property is False.

Use the AutoSize, FixedCellHeight and FixedCellWidth properties to defines the size of the control's cell. A cell displays a date. The FixedCellHeight and FixedCellWidth properties has effect only if the AutoSize is False. The following sample fixes the cell's size to (18,32):

```
CalendarCombo1.AutoSize = False
CalendarCombo1.FixedCellHeight = 18
CalendarCombo1.FixedCellWidth = 32
```

property CalendarCombo.FixedCellWidth as Long

Retrieves or sets a value that indicates the cell's width while the AutoSize is false.

Type	Description
Long	A long expression that defines the cell's width while the AutoSize property is False.

Use the AutoSize, FixedCellHeight and FixedCellWidth properties to defines the size of the control's cell. A cell displays a date. The FixedCellHeight and FixedCellWidth properties has effect only if the AutoSize is False. The following sample fixes the cell's size:

```
CalendarCombo1.AutoSize = False
CalendarCombo1.FixedCellHeight = 18
CalendarCombo1.FixedCellWidth = 32
```

property CalendarCombo.FocusIndex as Long

Specifies the index of the focused element.

Type	Description
Long	A long expression that indicates the index of the element being focused.

Use the FocusIndex property to specify the index of the element being focused. Use the Caption property to retrieve the caption of the element in the control's label. By default, the FocusIndex property is indicating the position where the day is displayed. Use the FocusIndex property to specify the index of element being focused. Use the FormatDate property to specify the predefined format of the date being displayed. Use the FormatUserDate property to specify the custom format of the date being displayed. The control fires the FocusIndexChanged event when the FocusIndex property is changed.

property CalendarCombo.Font as IFontDisp

Retrieves or sets the font to display the drop down portion of the control.

Type	Description
IFontDisp	A Font object that defines the font for the drop down portion of the control.

Defines the drop down portion's font. Use the LabelFont property to specify the label's font. Use the Bold, Italic, UnderLine or StrikeOut property to change the font attributes for a particular date.

property CalendarCombo.ForeColor as Color

Retrieves or sets the control's foreground color.

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

Defines the control's foreground color. Use the Background(exDropDownForeColor) property to specify the foreground color of the drop down portion of the control. Use the ForeColor property of Event object to define the foreground color for a particular date. Use the ForeColorDisabled property to specify the color to show the selected date when the checkbox is unchecked.

property CalendarCombo.ForeColorDisabled as Color

Retrieves or sets the control's foreground color when the date is locked.

Type	Description
Color	A Color expression that specifies the color to show the date in the control's label when the checkbox is unchecked.

By default, the ForeColorDisabled property is gray. It specifies the color to show the selected date, if the checkbox is unchecked. The property has effect if the AllowCheckBox property is True, and the check box is unchecked. Use the ForeColor property to specify the color of the selected date when check box is checked. The selected date is not shown, if the check box is unchecked, and the ForeColorDisabled property is not equal with the control's BackColor property, so the ForeColorDisabled can be used to hide the selected date when the checkbox is unchecked.

method CalendarCombo.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 CalendarCombo.

The FormatABC method formats the A,B,C values based on the giving expression and returns the result.

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 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

This property/method supports predefined constants and operators/functions as described here.

property CalendarCombo.FormatDate as FormatDateEnum

Retrieves or sets a value that indicates the format of date displayed.

Type	Description
FormatDateEnum	A FormatDateEnum expression that defines the format of the displayed date.

Use the FormatDate to change the format of the displayed date. The "ShortDate" displays date like: 12/31/1971, the "LongDate" displays the date like: "December 31 1971, Friday", and the "UserDate" format allows user to personalize how the date is displayed. Use the FormatUserDate property to specify the custom format when the FormatDate property is UserDate. Use the Caption property to retrieve the caption being displayed in the control's label.

For instance the following sample shows how to display the date like: 31 - 12 - 1971:

```
CalendarCombo1.FormatDate = UserDate
CalendarCombo1.FormatUserDate = "%d - %m - %y"
```

property CalendarCombo.FormatUserDate as String

Retrieves or sets a value that indicates the string format of the date while FormatDate is UserDate.

Type	Description
String	A string expression that defines the string format used to display the selected date, while the FormatDate property is UserDate

Use the FormatUserDate property to display date and time as you need. The FormatUserDate property has effect only if the FormatDate property is UserDate. Use the Caption property to retrieve the caption being displayed in the control's label. Use the SelDate or Value property to specify the date being displayed in the control's label. The AllowSpin property indicates whether the control displays a spin control to allow user changes the selected field using up or down arrows. Use the AllowCheckBox property to simulate as the browsed date is available or disable.

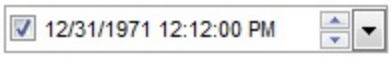
The FormatUserDate property should be a combination of text and one or more of the followings indicators:

- **%m** - indicates the month's number (1 to 12)
- **%mm** - indicates the month's number, in two digits (01 to 12)
- **%M** - indicates the month's name (The MonthNames property indicates the name of the months)
- **%M1** - indicates the first letter of the month's name
- **%M2** - indicates the first two letters of the month's name
- **%M3** - indicates the first three letters of the month's name
- **%d** - indicates the day number (1 to 31, depends on the month)
- **%dd** - indicates the day number, in two digits (01 to 31, depends on the month)
- **%y** - indicates the year number,
- **%w** - indicates the day of the week
- **%w1** - indicates the first letter of the day of the week
- **%w2** - indicates the first two letters of the day of the week
- **%w3** - indicates the first three letters of the day of the week
- **%h** - indicates the hour (1 to 12, if the %a is present, 0 to 23 if the %a is missing)
- **%hh** - indicates the hour in two digits (01 to 12, if the %a is present, 00 to 23 if the %a is missing)
- **%n** - indicates the minute (0 to 59)
- **%nn** - indicates the minute in two digits (00 to 59)
- **%s** - indicates the second (0 to 59)
- **%ss** - indicates the second in two digits (00 to 59)

- **%a** - indicates the AM/PM field. (AM or PM). Use the AMPM property to specifies the AM/PM indicates in %a fields.

The following VB sample displays date and time like: "12/31/1971 12:12:00 PM"

```
With CalendarCombo1
   .FormatDate = UserDate
   .FormatUserDate = "%m/%d/%y %hh:%nn:%ss %a"
End With
```



The following VB sample displays date like: "1971/12/31"

```
With CalendarCombo1
   .FormatDate = UserDate
   .FormatUserDate = "%y/%m/%d"
End With
```

method CalendarCombo.FreezeEvents (Freeze as Boolean)

Prevents the control to fire any event.

Type	Description
Freeze as Boolean	A Boolean expression that specifies whether the control' events are froze or unfroze

The FreezeEvents(True) method freezes the control's events until the FreezeEvents(False) method is called.

property CalendarCombo.GridLineColor as Color

Retrieves or sets a value that indicates the grid lines color.

Type	Description
Color	A color expression that indicates the grid lines color.

Use the DrawGridLine property to hide the grid lines.

property CalendarCombo.HeaderBackColor as Color

Retrieves or sets a value that indicates the background color used for weeks and week days headers.

Type	Description
Color	A color expression a value that indicates the background color used for weeks and week days headers.

Use the HeaderForeColor property to change the foreground color for weeks and week days headers. Use the ShowWeeks to hide the weeks header. Use the ShowDays to hide the week days header.

property CalendarCombo.HeaderForeColor as Color

Retrieves or sets a value that indicates the foreground color used for weeks and week days headers.

Type	Description
Color	A color expression that indicates the week and week day's foreground color

Use the HeaderBackColor property to change the foreground color for weeks and week days headers. Use the ShowWeeks to hide the weeks header. Use the ShowDays to hide the week days header.

property CalendarCombo.HideDropDownButton as Long

Determines whether the drop down button is visible or hidden when the control loses the focus.

Type	Description
Long	A long expression that specifies whether the control's drop down button is visible or hidden. Possible values are 0, -1 and 1.

By default, the HideDropDownButton property is 0, which makes the drop down button being always visible. The DropDown property specifies whether the control's drop down portion of the control is visible or hidden. Use the Value property to change the control's date. Use the AllowSpin property to display a spin control inside the control's label. Use the Background(exDropDownButtonUp) property to change the visual appearance for the drop down button of the control.

The HideDropDownButton property can be

- 0, the drop down button is always visible
- 1, the drop down button is always hidden
- -1, the drop down button is visible if the control gains the focus, and it is hidden as soon as the control loses the focus.

property CalendarCombo.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

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

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, tooltips, comments, 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).

property CalendarCombo.hWnd as Long

Retrieves the control's window handle.

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

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

method CalendarCombo.Images (Handle as Variant)

Sets the control's handle image list.

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 control's images panel. Use the ShowImageList property to hide the control's images panel. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the Image property to assign an icon to a date. Use the CheckImage property to change the visual appearance of the control's checkbox.

The following sample loads icons from a BASE64 encoded strings:

```
With Calendar1
   .BackColor = vbWhite
   .AutoSize = False
   .FixedCellWidth = 32
   .Images
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExm

   With .Events.Add(Date + 1)
      .Image = 1
   End With
   With .Events.Add(Date + 2)
      .Image = 2
   End With
End With
```

If you run the sample you will see:

The following sample uses the Microsoft Image List control:

```
CalendarCombo1.Images ImageList1.hImageList
```


Calendar...
Images
Drag here an executable file,
a dynamic-link library, an
icon file or a regular file

property CalendarCombo.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 CalendarCombo.IndexFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

Retrieves the index of the date's element from the point.

Type	Description
X as OLE_XPOS_PIXELS	A long expression that indicates the x-coordinate of the point.
Y as OLE_YPOS_PIXELS	A long expression that indicates the y-coordinate of the point.
Long	A long expression that indicates the index of the date's element from the point.

The IndexFromPoint property retrieve the index of the element from the cursor. If the x and y parameters are -1, the IndexFromPoint property retrieves the index of the element from the current cursor position. Use the FocusIndex property to specify the index of the date's element that gets the focus. Use the Caption property to retrieve the caption of the date's element.

The following VB sample prints the date's element being double clicked:

```
Private Sub CalendarCombo1_DblClick(Shift As Integer, X As Single, Y As Single)
  With CalendarCombo1
    h = .IndexFromPoint(-1, -1)
    Debug.Print .Caption(h)
  End With
End Sub
```

The following VB.NET sample prints the date's element being double clicked:

```
Private Sub AxCalendarCombo1_DblClick(ByVal sender As Object, ByVal e As
AxEXCALENDARLib._ICalendarComboEvents_DblClickEvent) Handles
AxCalendarCombo1.DblClick
  With AxCalendarCombo1
    System.Diagnostics.Debug.Print(.get_Caption(.get_IndexFromPoint(e.x, e.y)))
  End With
End Sub
```

The following C# sample prints the date's element being double clicked:

```
private void axCalendarCombo1_DblClick(object sender,
AxEXCALENDARLib._ICalendarComboEvents_DblClickEvent e)
{

System.Diagnostics.Debug.Print(axCalendarCombo1.get_Caption(axCalendarCombo1.get_I
 e.y)));
}
```

The following C++ sample prints the date's element being double clicked:

```
void OnDblClickCalendarcombo1(short Shift, long X, long Y)
{
    long i = m_calendarcombo.GetIndexFromPoint( X, Y );
    OutputDebugString( m_calendarcombo.GetCaption( COleVariant( i ) ) );
}
```

The following VFP sample prints the date's element being double clicked:

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

with thisform.CalendarCombo1
    ?.Caption(.IndexFromPoint(x,y))
endwith
```

property CalendarCombo.LabelBounds as Variant

Specifies the bounds to display the control's label.

Type	Description
Variant	A VARIANT expression that indicates a safe array of 4 elements (VT_I4), that indicates left, top, right and bottom of coordinates of the control's label.

Use the LabelBounds property to get the coordinates of the control's label, so you can replace the static label with a mask edit control or any other control. The /NET or /WPF version provides the LabelBoundsRect property that gets directly the client rectangle of the control's label. The value of the LabelBoundsRect property can be passed to the Bounds of the inside control, so it covers the control's static label. The C:\Program Files\Exontrol\ExCalendar.NET\Sample\VB.NET\Sample.Exontrol.Date-Picker sample shows you can you can use the LabelBounds or LabelBoundsRect property.

property CalendarCombo.LabelFont as IFontDisp

Retrieves or sets the label's font.

Type	Description
IFontDisp	A Font object that defines the label's font.

The LabelFont property defines the font for the control's label. The Font property fefines the drop down portion's font. The LabelHeight property specifies the height of the label in pixels. Use the Bold, Italic, UnderLine or StrikeOut property to change the font attributes for a particular date.

property CalendarCombo.LabelHeight as Long

Specifies the label's height.

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

Use the LabelHeight property to specify the height of the control's label. The LabelFont property defines the font for the control's label.

property CalendarCombo.LastVisibleDate as Date

Retrieves the last visible date.

Type	Description
Date	A DATE expression that specified the last visible date in the calendar.

The LastVisibleDate property retrieves the last visible date being displayed in the calendar. Use the ShowNonMonthDays property to display the dates that are not part of the month. Use the FirstVisibleDate property to get the first visible date being displayed in the calendar.

property CalendarCombo.Locked as Boolean

Specifies whether the user can change the selection.

Type	Description
Boolean	A boolean expression that specifies whether the control is locked or unlocked.

By default, the Locked property is False. Use the Locked property to lock the control. While the control is locked, the drop down portion of the control can be displayed, while the control is disabled, the control can't display it's drop down portion of the control. While locked the user can't change the selected date.

property CalendarCombo.MarkToday as Boolean

Retrieves or sets a value that indicates whether the control marks the today date.

Type	Description
Boolean	A boolean expression that indicates whether the control marks the today date.

Use the MarkToday property to mark today date. By default, the MarkToday property is False.

property CalendarCombo.MaskOnEmpty as String

Specifies the masking string for each entity when the date is empty.

Type	Description
String	A String expression that indicates the masking string when the SelDate property is empty. For instance, if the control's label displays the date such as 6/5/2013, if the Value property is set on 0, the control's label displays actually _/_/_ caption.

By default, the MaskOnEmpty property is "_". Use the MaskOnEmpty property to specify the caption of the control's label when the SelDate/Value property is empty (0/null date). If the MaskOnEmpty property is "" (empty string), the control's label displays nothing when the SelDate/Value property is 0.

The following VB sample displays nothing when user presses the Delete key:

```
Private Sub CalendarCombo1_KeyDown(KeyCode As Integer, Shift As Integer)
   If (KeyCode = vbKeyDelete) Then
      CalendarCombo1.Value = 0
   End If
End Sub

Private Sub Form_Load()
   With CalendarCombo1
      .AllowCheckBox = exHidden
      .MaskOnEmpty = ""
   End With
End Sub
```

The following VB/NET sample displays nothing when user presses the Delete key:

```
Public Class Form1
   Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
      With Excalendarcombo1
         .AllowCheckBox = exontrol.EXCALENDARLib.VisibleEnum.exHidden
         .MaskOnEmpty = ""
      End With
```

```
    End Sub

    Private Sub Excalendarcombo1_KeyDown(ByVal sender As Object, ByRef KeyCode As
Short, ByVal Shift As Short) Handles Excalendarcombo1.KeyDown
        With Excalendarcombo1
            If (KeyCode = Keys.Delete) Then
                .Value = Nothing
            End If
        End With
    End Sub
End Class
```

property CalendarCombo.MaxDate as Date

Retrieves or sets the min date.

Type	Description
Date	A Date expression that indicates the upper limit for dates to be displayed or selected.

By default, the MaxDate property is Dec 31, 9999. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected. The Date property is automatically updated so the browsed date fits the limited range. When the browsed date is changed the control fires the DateChanged event. By default, the Date property points to the current date.

property CalendarCombo.MaxMonthX as Long

Specifies the maximum number of months horizontally displayed.

Type	Description
Long	A long expression that specifies the maximum number of months horizontally displayed.

By default, the MaxMonthX property is 6. Use the MaxMonthX property to define the maximum number of months displayed horizontally. Use the MinMonthX property to define the minimum number of months displayed horizontally. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property CalendarCombo.MaxMonthY as Long

Specifies the maximum number of months vertically displayed.

Type	Description
Long	A long expression that defines the maximum number of months vertically displayed.

By default, the the MaxMonthY property is 1. Use the MaxMonthY property to define the maximum number of months vertically displayed. Use the MinMonthY property to define the minimum number of months vertically displayed. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property CalendarCombo.MaxScrollYear as Long

Specifies the maximum year when scrolling.

Type	Description
Long	A long expression that indicates the maximum year when user scrolls the calendar.

Use the MinScrollYear and MaxScrollYear properties to specify the range of dates where user is allowed to scroll the calendar.

property CalendarCombo.MinDate as Date

Retrieves or sets the min date.

Type	Description
Date	A Date expression that indicates the lower limit for dates to be displayed or selected.

By default, the MinDate property is Jan 1, 100. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected. The Date property is automatically updated so the browsed date fits the limited range. When the browsed date is changed the control fires the DateChanged event. By default, the Date property points to the current date.

property CalendarCombo.MinMonthX as Long

Specifies the minimum number of months horizontally displayed.

Type	Description
Long	A long expression that specifies the minimum number of months horizontally displayed.

By default, the MinMonthX property is 1. Use the MinMonthX property to define the minimum number of months displayed horizontally Use the MaxMonthX property to define the maximum number of months displayed horizontally. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property CalendarCombo.MinMonthY as Long

Specifies the minimum number of months vertically displayed.

Type	Description
Long	A long expression that specifies the minimum number of months vertically displayed.

By default, the the MaxMonthY property is 1. Use the MinMonthY property to define the minimum number of months vertically displayed. Use the MaxMonthY property to define the maximum number of months vertically displayed. Use the MinDate and MaxDate property to specify a range to limit the date to be displayed or selected.

property CalendarCombo.MinScrollYear as Long

Specifies the minimum year when scrolling.

Type	Description
Long	A long expression that indicates the minimum year when user scrolls the calendar.

Use the MinScrollYear and MaxScrollYear properties to specify the range of dates where user is allowed to scroll the calendar.

property CalendarCombo.MonthName(Month as MonthEnum) as String

Retrieves or sets the month's name.

Type	Description
Month as MonthEnum	A MonthEnum expression that indicates the month
String	A String expression that specifies the month's name.

Use the MonthNames property to change the names for all months

property CalendarCombo.MonthNames as String

Retrieves or sets a value that indicates the list of month names, separated by space.

Type	Description
String	A string expression hat indicates the list of month names, separated by space.

By default, the MonthNames is "January February March April May June July August September October November December". For instance, for French you have to use somethign like: "Janvier Février Mars Avril Mai Juin Juillet Aoűt Septembre Octobre Novembre Décembre". Use the FirstDay property to change the first day of the week. Use the WeekNames property to define the name for each week day

property CalendarCombo.NonMonthDaysColor as Color

Retrieves or sets a value that indicates the color to show the non-month days.

Type	Description
Color	A Color expression that specifies the color to show the days that are not owned by the current month.

By default, the NonMonthDaysColor property is gray. The NonMonthDaysColor property specifies the foreground color to show the days that are not owned by displaying month. The ShowNonMonthDays property specifies whether the calendar displays the days that are not-owned by the displaying month. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date. The NonworkingDaysForeColor property specifies the foreground color for non-working days.

The following screen shot shows the non-month days in red:

property CalendarCombo.NonworkingDays as Long

Retrieves or sets a value that indicates the non-working days, for each week day a bit.

Type	Description
Long	A long expression that indicates the non-working days in a week.

By default, the NonworkingDays property is 65. The last significant byte in the NonworkingDays expression has the following meaning:

-	Sa	Fr	Th	We	Tu	Mo	Su
0	X	X	X	X	X	X	X
128	64	32	16	8	4	2	1

where **X** could be 1 (nonworking day) or 0 (working day), **Sa** means Saturday, **Fr** means Friday, and so on. For instance, the 65 value means Saturday and Sunday are non-working days. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days. For instance, if the NonworkingDaysPattern is exPatternEmpty the non-working days are not highlighted. Use the ShowNonMonthDays property to specify whether the dates that are not part of the month are visible or hidden. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date. The NonworkingDaysForeColor property specifies the foreground color for non-working days. The FirstDay property specifies the first day of the week.

The following VB sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
With CalendarCombo1
   .NonworkingDays = 2 ^ (EXCALENDARLibCtl.Sunday - 1) Or 2 ^
(EXCALENDARLibCtl.Monday - 1)
End With
```

The following C++ sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
m_calendarcombo.SetNonworkingDays( 1 << ( EXCALENDARLib::Sunday - 1 ) | 1 << ( EXCALENDARLib::Monday - 1 ) );
```

The following VB.NET sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
With AxCalendarCombo1
    .NonworkingDays = 2 ^ (EXCALENDARLib.WeekDayEnum.Sunday - 1) Or 2 ^ (EXCALENDARLib.WeekDayEnum.Monday - 1)
End With
```

The following C# sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
axCalendarCombo1.NonworkingDays = 1 << (Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Sunday) - 1) | 1 << (Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Monday) - 1);
```

The following VFP sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
with thisform.CalendarCombo1
    .NonworkingDays = 2 ^ 0 + 2 ^ 1
endwith
```

property CalendarCombo.NonworkingDaysColor as Color

Retrieves or sets a value that indicates the color to fill the non-working days.

Type	Description
Color	A Color expression that indicates the color to fill the non-working days.

Use the NonworkingDaysColor property to specify the color being used by the NonworkingDaysPattern property. Use the NonworkingDays property to specify the nonworking days in a week. Use the NonworkingDaysPattern property to specify the pattern to fill the non-working days. For instance, if the NonworkingDaysPattern is exPatternEmpty the non-working days are not highlighted. The NonworkingDaysForeColor property specifies the foreground color for non-working days. The FirstDay property specifies the first day of the week.

The following VB sample marks Sunday and Monday days on red:

```
With CalendarCombo1
   .NonworkingDays = 2 ^ (EXCALENDARLibCtl.Sunday - 1) Or 2 ^
(EXCALENDARLibCtl.Monday - 1)
   .NonworkingDaysColor = RGB(255, 0, 0)
End With
```

The following C++ sample sample marks Sunday and Monday days on red:

```
m_calendarcombo.SetNonworkingDays( 1 << ( EXCALENDARLib::Sunday - 1 ) | 1 << (
EXCALENDARLib::Monday - 1 ) );
m_calendarcombo.SetNonworkingDaysColor( RGB(255,0,0,) );
```

The following VB.NET sample marks Sunday and Monday days on red:

```
With AxCalendarCombo1
   .NonworkingDays = 2 ^ (EXCALENDARLib.WeekDayEnum.Sunday - 1) Or 2 ^
(EXCALENDARLib.WeekDayEnum.Monday - 1)
   .NonworkingDaysColor = Color.Red
End With
```

The following C# sample marks Sunday and Monday days on red:

```
axCalendarCombo1.NonworkingDays = 1 <<
```

```
(Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Sunday) - 1) | 1 <<
(Convert.ToInt32(EXCALENDARLib.WeekDayEnum.Monday) - 1);
axCalendarCombo1.NonworkingDaysColor = Color.Red;
```

The following VFP sample sample marks Sunday and Monday days on red:

```
with thisform.CalendarCombo1
   .NonworkingDays = 2 ^ 0 + 2 ^ 1
   .NonworkingDaysColor = RGB(255,0,0)
endwith
```

property CalendarCombo.NonworkingDaysForeColor as Color

Retrieves or sets a value that indicates the foreground color for non-working days.

Type	Description
Color	A Color expression that specifies the color to show the non-working days of the month.

By default, the NonworkingDaysForeColor property is black. The NonworkingDaysForeColor property specifies the foreground color for non-working days. The NonworkingDaysColor property specifies the color to show the pattern for non-working days. Use the NonworkingDays property to specify the nonworking days in a week. The FirstDay property specifies the first day of the week. Use the NonworkingDaysPattern property to specify the pattern to fill the non-working days. Use the NonMonthDaysColor property to specify the color to show days that are not owned by displayed month. By default, the ForeColor property specifies the color to show the days.

The following screen shot shows the non-working days in red:

property CalendarCombo.NonworkingDaysPattern as PatternEnum

Retrieves or sets a value that indicates the pattern being used to fill non-working days.

Type	Description
PatternEnum	A PatternEnum expression that indicates the pattern to fill non working days.

Use the NonworkingDaysPattern property to specify the pattern to fill non-working days. By default, the NonworkingDaysPattern property is exPatternDot. If the NonworkingDaysPattern property is exPatternEmpty, the non-working days are not highlighted. Use the NonworkingDays property to specify the non-working days in a week. The NonworkingDaysColor property specifies the color being used to fill the non-working days. The NonworkingDaysForeColor property specifies the foreground color for non-working days. The FirstDay property specifies the first day of the week.

The following VB sample draws non-working days using the exPatternShadow brush:

```
With CalendarCombo1
   .NonworkingDaysPattern = exPatternShadow
End With
```

The following C++ sample draws non-working days using the exPatternShadow brush:

```
m_calendarcombo.SetNonworkingDaysPattern( 3 /*exPatternShadow*/ );
```

The following VB.NET sample draws non-working days using the exPatternShadow brush:

```
With AxCalendarCombo1
   .NonworkingDaysPattern = EXCALENDARLib.PatternEnum.exPatternShadow
End With
```

The following C# sample draws non-working days using the exPatternShadow brush:

```
axCalendarCombo1.NonworkingDaysPattern =
EXCALENDARLib.PatternEnum.exPatternShadow
```

The following VFP sample draws non-working days using the exPatternShadow brush:

```
with thisform.CalendarCombo1
    .NonworkingDaysPattern = 3
endwith
```

property CalendarCombo.Picture as IPictureDisp

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

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

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the Background(exDropDownBackColor) property to specify the background color of the drop down portion of the control. You can use the Picture property to add your logo on the control's background. The picture is displayed on the drop down portion of the control. The /NET version provides the BackgroundImage property.

The following screen shot show a picture on the control's background:

property CalendarCombo.PictureDisplay as PictureDisplayEnum

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

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

By default, the PictureDisplay property is exTile. The PictureDisplay property specifies how the Picture is displayed on the control's background. If the control has no picture associated the PictureDisplay property has no effect. Use the Background(exDropDownBackColor) property to specify the background color of the drop down portion of the control. The /NET version provides the BackgroundImageLayout property.

property CalendarCombo.ScrollOnDrop as Boolean

Specifies a value that indicates whether the drop down calendar is scrolled when it is shown.

Type	Description
Boolean	A boolean expression that indicates whether the drop down calendar is scrolled when it is shown.

By default, the ScrollOnDrop property is True.

property CalendarCombo.SelBackColor as Color

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

Type	Description
Color	A color expression that specifies the selection background color.

Use the SelBackColor and SelForeColor properties to define the colors for selected date(s) in the control's label. Use the Background(exDropDownSelBackColor) property to specify the background color of the drop down portion of the control.

The following VB sample changes the selection colors for label and drop down part of the control:

```
With CalendarCombo1
   .SelBackColor = RGB(255,0,0)
   .SelForeColor = RGB(255,255,255)
   .Background(exDropDownSelBackColor) = .SelBackColor
   .Background(exDropDownSelForeColor) = .SelForeColor
End With
```

The following VB.NET sample changes the selection colors for label and drop down part of the control (assembly version):

```
With Excalendarcombo1
   .SelBackColor = Color.FromArgb(255,0,0)
   .SelForeColor = Color.FromArgb(255,255,255)

.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDropDownSelBackColo

.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDropDownSelForeColo

End With
```

The following C# sample changes the selection colors for label and drop down part of the control (assembly version):

```
excalendarcombo1.SelBackColor = Color.FromArgb(255,0,0);
```

```
excalendarcombo1.SelForeColor = Color.FromArgb(255,255,255);
excalendarcombo1.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDrop

excalendarcombo1.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDrop
```

property CalendarCombo.SelDate as Date

Selects a date.

Type	Description
Date	A DATE expression that is selected

The SelDate property indicates the selected date. The Value property is identical with the SelDate property. The Date property specifies the date being browsed in the drop down portion of the control. Use the MaskOnEmpty property to specify the caption being displayed in the control's label when the SelDate property is empty. The SelectionChanged event is fired if the user changes the browsed date. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date.

In VB/NET or C# you may need to use the Nothing, null or Date.FromOADate(0) to convert the 0 value to a null date.

For instance,

- *SelDate = Nothing* or *SelDate = Date.FromOADate(0)* makes the control to display no date, equivalent with SelDate = 0, in VB6
- *If (.SelDate.ToOADate() = 0) Then*, checks if the control's SelDate is empty, equivalent with if (SelDate = 0) then , in VB6

The following VB sample set the SelDate property on empty, when the user presses the Delete key: (displays an empty date)

```
Private Sub CalendarCombo1_KeyDown(KeyCode As Integer, Shift As Integer)
  If (KeyCode = vbKeyDelete) Then
    CalendarCombo1.SelDate = 0
  End If
End Sub
```

or

```
Private Sub CalendarCombo1_KeyDown(KeyCode As Integer, Shift As Integer)
  If (KeyCode = vbKeyDelete) Then
    CalendarCombo1.Value = 0
  End If
End Sub
```

or

```
Private Sub CalendarCombo1_KeyDown(KeyCode As Integer, Shift As Integer)
   If (KeyCode = vbKeyDelete) Then
      CalendarCombo1 = 0
   End If
End Sub
```

The following VB sample changes the initialization date, when the user presses a key, while the control displays an empty date:

```
Private Sub CalendarCombo1_KeyPress(KeyAscii As Integer)
   If (CalendarCombo1 = 0) Then
      CalendarCombo1 = "31/12/1971"
   End If
End Sub
```

or

```
Private Sub CalendarCombo1_KeyPress(KeyAscii As Integer)
   If (CalendarCombo1 = 0) Then
      CalendarCombo1.SelDate = "31/12/1971"
   End If
End Sub
```

or

```
Private Sub CalendarCombo1_KeyPress(KeyAscii As Integer)
   If (CalendarCombo1 = 0) Then
      CalendarCombo1.Value = "31/12/1971"
   End If
End Sub
```

property CalendarCombo.SelForeColor as Color

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

Type	Description
Color	A color expression that indicates the selection foreground color.

Use the SelForeColor and SelBackColor properties to define the foreground and background colors of the selected date. Use the Background(exDropDownSelForeColor) property to specify the foreground color of the drop down portion of the control.

The following VB sample changes the selection colors for label and drop down part of the control:

```
With CalendarCombo1
   .SelBackColor = RGB(255,0,0)
   .SelForeColor = RGB(255,255,255)
   .Background(exDropDownSelBackColor) = .SelBackColor
   .Background(exDropDownSelForeColor) = .SelForeColor
End With
```

The following VB.NET sample changes the selection colors for label and drop down part of the control (assembly version):

```
With Excalendarcombo1
   .SelBackColor = Color.FromArgb(255,0,0)
   .SelForeColor = Color.FromArgb(255,255,255)

.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDropDownSelBackColo

.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDropDownSelForeColo

End With
```

The following C# sample changes the selection colors for label and drop down part of the control (assembly version):

```
excalendarcombo1.SelBackColor = Color.FromArgb(255,0,0);
```

```
excalendarcombo1.SelForeColor = Color.FromArgb(255,255,255);
excalendarcombo1.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDrop

excalendarcombo1.set_Background(exontrol.EXCALENDARLib.BackgroundPartEnum.exDrop
```

property CalendarCombo.ShowDays as Boolean

Retrieves or sets a value that indicates whether the week days header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the week days header is visible or hidden.

Use the ShowDays property to hide the week days header. Use the HeaderBackColor and HeaderForeColor properties to change the background and foreground colors of the week days header.

property CalendarCombo.ShowFocusRect as Boolean

Specifies whether the focus rectangle is shown around the label while the control has the focus.

Type	Description
Boolean	A Boolean expression that specifies whether the control's label shows a rectangle while the control has the focus.

By default, the ShowFocusRect property is True. Use the ShowFocusRect property on False, to hide the rectangle drawing around the control's label while it has the focus.

property CalendarCombo.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.

The control's images panel is visible only at design time. By default, the ShowImageList property is True. Use the Images method to assign a list of icons to the control, at run time.

The following screen shot shows the control's images panel, available only at design time:

property CalendarCombo.ShowMonth as Boolean

Retrieves or sets a value that indicates whether the month header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the month header is visible or hidden.

Use the ShowMonth property to hide the month's header.

property CalendarCombo.ShowMonthSelector as Boolean

Retrieves or sets a value that indicates whether the user is able to select a new month by clicking in the month header.

Type	Description
Boolean	A boolean expression that indicates whether the user is able to select a new month by clicking in the month header.

By default, the ShowMonthSelector property is True. Use the ShowYearSelector property to specify whether the selector for years are visible or hidden. The control DateChanging event when a selector is clicked.

- If the ShowMonthSelector and ShowYearSelector property are True, the Left and Right selectors changes the current year to prev or next year
- If the ShowMonthSelector is False and ShowYearSelector property is True, the Left and Right selectors changes the current month to prev or next month

The following screen shot shows the selectors (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the month selector once the user clicks the month header (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the selectors (the left and right selectors changes the current month to previous or next. no month selector is displayed when the user clicks the month header, **ShowMonthSelector(False), ShowYearSelector(True)**)

property CalendarCombo.ShowNonMonthDays as Boolean

Specifies whether the control displays the dates that are not part of the month.

Type	Description
Boolean	A boolean expression that indicates whether the date that are not part of the month are visible or hidden.

By default, the ShowNonMonthDays property is True. Use the ShowNonMonthDays property to hide the the dates that are not part of the month. Use the NonWorkingDays property to specify the non working days. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date.

The following screen shot shows the control when the ShowNonMonthDays property is False:

The following screen shot shows the control when the ShowNonMonthDays property is True:

property CalendarCombo.ShowTodayButton as Boolean

Retrieves or sets a value that indicates whether the today button is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the today button is visible or hidden.

By clicking Today button, the Date property set to today date. Use the ShowTodayButton to show or hide the today button.

property CalendarCombo.ShowWeeks as Boolean

Retrieves or sets a value that indicates whether the weeks header is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the weeks header is visible or hidden.

Use the ShowWeeks property to show the weeks header. The weeks header displays the week number into the year. Use the ShowDays property to hide the week days header. Use the HeaderBackColor and HeaderForeColor properties to change the background and foreground colors of the weeks header. Use the ShowNonMonthDays property to specify whether the dates that are not part of the month are visible or hidden.

property CalendarCombo.ShowYearScroll as Boolean

Retrieves or sets a value that indicates whether the scroll bar for changing the year is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the scroll bar for changing the year is visible or hidden

By default, the ShowYearScroll property is True.

property CalendarCombo.ShowYearSelector as Boolean

Retrieves or sets a value that indicates whether the year selector is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the year selector is visible or hidden.

By default, the ShowYearSelector property is True. Use the ShowMonthSelector property to specify whether the selector for month are visible or hidden. The control DateChanging event when a selector is clicked.

- If the ShowMonthSelector and ShowYearSelector property are True, the Left and Right selectors changes the current year to prev or next year
- If the ShowMonthSelector is False and ShowYearSelector property is True, the Left and Right selectors changes the current month to prev or next month

The following screen shot shows the selectors (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the month selector once the user clicks the month header (by default, the left and right selectors changes the year to previous or next year, while the up and down arrow changes the current month to previous or next. Clicking the month on the header, makes the control to display a list of month to select from, **ShowMonthSelector(True), ShowYearSelector(True)**)

The following screen shot shows the selectors (the left and right selectors changes the current month to previous or next. no month selector is displayed when the user clicks the month header, **ShowMonthSelector(False), ShowYearSelector(True)**)

property CalendarCombo.Template as String

Specifies the control's template.

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

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

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.*

method CalendarCombo.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 CalendarCombo.TodayCaption as String

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

Type	Description
String	A string expression that indicates the caption for combo's today button.

Use the TodayCaption property to specify the caption for combo's today button. Use the ShowTodayButton property to hide the control's today button. By default the TodayCaption property is "Today" .

property CalendarCombo.ToolTipDelay as Long

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

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

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

property CalendarCombo.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

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

property CalendarCombo.ToolTipPopDelay as Long

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

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

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

property CalendarCombo.ToolTipWidth as Long

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

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

Use the ToolTipWidth property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipFont property to assign a font for the control's tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

property CalendarCombo.UseVisualTheme as UIVisualThemeEnum

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

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

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

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

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

property CalendarCombo.Value as Date

Retrieves or sets the browsed date. Ensures that the date is visible.

Type	Description
Date	A date expression that indicates the selected date.

The Value property (default property of the control) indicates the selected date. The Value property is identical with the SelDate property. The Date property specifies the date being browsed in the drop down portion of the control. Use the MaskOnEmpty property to specify the caption being displayed in the control's label when the SelDate property is empty. The SelectionChanged event is fired if the user changes the browsed date. Use the AllowCheckBox property to display a check box left to the date. Use the AllowCheckBox property to display a check box left to the date, so a date can be selected or not as follows, If the Value property is not zero, the check box is checked, else the checkbox is unchecked, and the labels shows grayed.

In VB/NET or C# you may need to use the Nothing, null or Date.FromOADate(0) to convert the 0 value to a null date.

For instance,

- *Value = Nothing* or *Value = Date.FromOADate(0)* makes the control to display no date, equivalent with Value = 0, in VB6
- *If (.Value.ToOADate() = 0) Then*, checks if the control's Value is empty, equivalent with if (Value = 0) then , in VB6

The following VB sample set the SelDate property on empty, when the user presses the Delete key: (displays an empty date)

```
Private Sub CalendarCombo1_KeyDown(KeyCode As Integer, Shift As Integer)
  If (KeyCode = vbKeyDelete) Then
    CalendarCombo1.SelDate = 0
  End If
End Sub
```

or

```
Private Sub CalendarCombo1_KeyDown(KeyCode As Integer, Shift As Integer)
  If (KeyCode = vbKeyDelete) Then
    CalendarCombo1.Value = 0
  End If
```

```
End Sub
```

property CalendarCombo.VisualAppearance as Appearance

Retrieves the control's appearance.

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

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

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

- drop down button for the CalendarCombo object, Background property
- months, weeks, days header, Background property, HeaderBackColor property,
- up down, left or right arrows, Background property
- selected date(s), SelBackColor property
- events, BackColor property
- Today button, scrolling dates area, Background property
- today date, MarkToday property
- selected items in the months selector, Background property

property CalendarCombo.WaitAutoAdvance as Long

Specifies the time in ms to wait until the selection moves to the next editing field in the CalendarCombo control.

Type	Description
Long	A Long expression that specifies the time in ms to wait until the selection moves to the next editing field in the CalendarCombo control.

By default, the WaitAutoAdvance property is 500 ms. Use the WaitAutoAdvance property to change the the time in ms to wait until the selection moves to the next editing field in the CalendarCombo control. The WaitAutoAdvance property has effect only if the AutoAdvance property is not exAdvanceNone.

property CalendarCombo.WeekDayName(WeekDay as WeekDayEnum) as String

Retrieves or sets a value that indicates the week day short name in the week days header.

Type	Description
WeekDay as WeekDayEnum	A WeekDayEnum expression that specifies the day in the week.
String	A String expression that specifies the name of the day in the week.

Use the WeekDayName property to change the name of a specified day of the week. Use the WeekDays property to assign a name for all days in the week.

property CalendarCombo.WeekDays as String

Retrieves or sets a value that indicates the list of short names for each week day, separated by space.

Type	Description
String	A string expression that indicates the list of short names for each week day, separated by space.

Event object

An Event object points to a DATE and contains graphical information like: colors, font attributes, images, tooltips, and so on. Use the Events property in order to access the Events collection

Name	Description
BackColor	Retrieves or sets a value that indicates the event's background color.
BackgroundExt	Indicates additional colors, text, images that can be displayed on the event's background using the EBN string format.
BackgroundExtValue	Specifies at runtime, the value of the giving property for specified part of the background extension.
Bold	Retrieves or sets a value that indicates whether the event should appear in bold.
Caption	Retrieves or sets a value that indicates the event's caption.
Comment	Retrieves or sets a value that indicates the event's comment.
CommentTitle	Retrieves or sets a value that indicates the event's comment title.
Date	Gets the event's date.
Disabled	Retrieves or sets a value that indicates whether the event is disabled or enabled.
ForeColor	Retrieves or sets a value that indicates the event's foreground color.
Image	Retrieves or sets the event's image.
Italic	Retrieves or sets a value that indicates whether the event should appear in italic.
Marker	Retrieves or sets a value that indicates whether the event's marker is visible or hidden.
Repetitive	Returns or sets the expression to determine the repetitive event.
StrikeOut	Retrieves or sets a value that indicates whether the event should appear in strikeout.
	Retrieves or sets a value that indicates whether the event

Underline	appears as underlined.
UserData	Retrieves or sets the event's user associated extra data.

property Event.BackColor as Color

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

Type	Description
Color	A color expression that indicates the date event'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.

Use the Events property to access the Events collection. Use the Background property to change the visual appearance for parts in the control. Use the Background(exDropDownAppearance) property to change the visual appearance of the drop down portion of the control. Use the Background(exDropDownBackColor) property to specify the background color of the drop down portion of the control. The BackgroundExt property provides unlimited options to show additional colors, text, icons, images, frames, patterns, ... to any event.

The following sample shows how to bold the "tomorrow" date:

```
Calendar1.Events.Add(Date() + 1).Bold = True
```

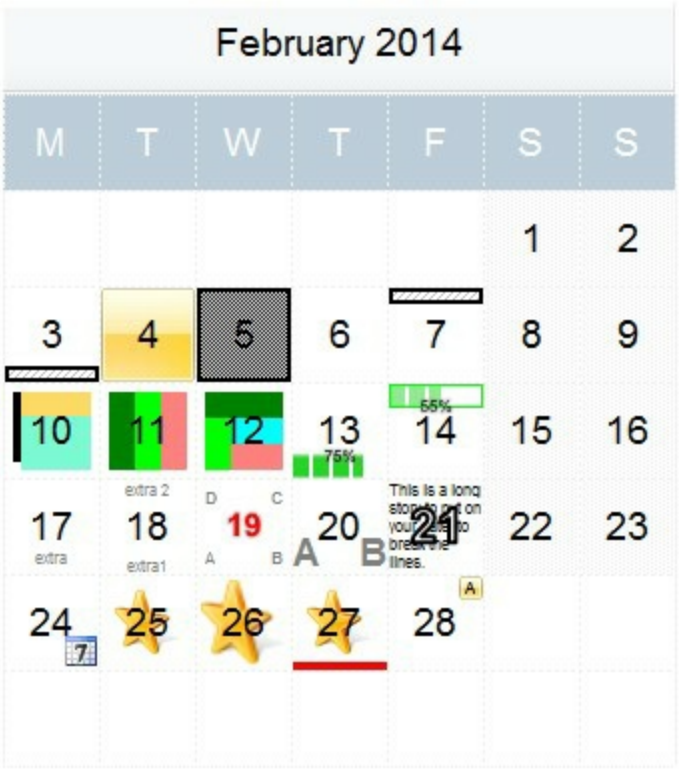
property Event.BackgroundExt as String

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

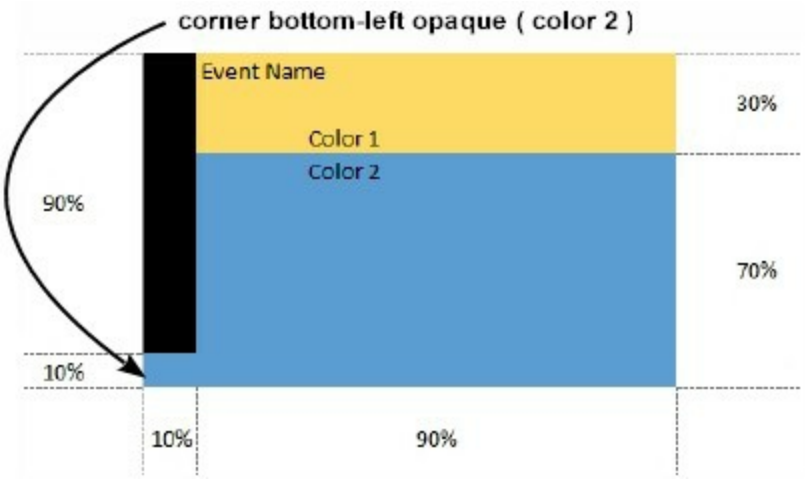
Type	Description
String	A String expression (**"EBN String Format"**) that defines the layout of the UI to be applied on the object's background. The syntax of EBN String Format in BNF notation is shown bellow. *You can use the EBN's Builder of eXButton/COM control to define visually the EBN String Format.*

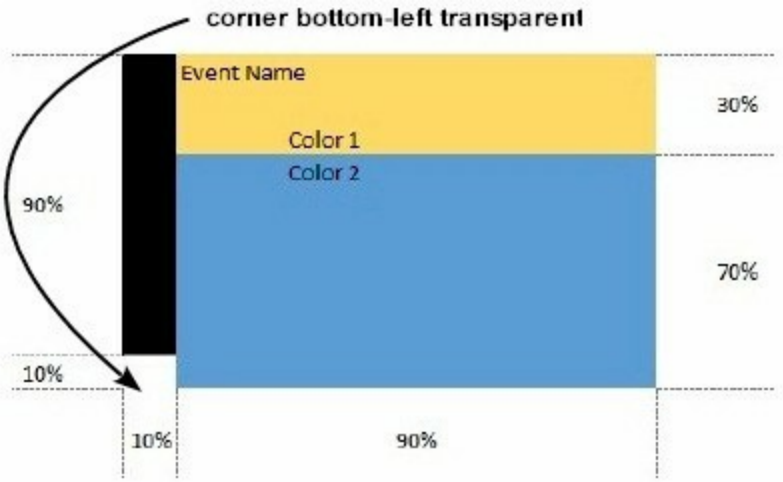
By default, the BackgroundExt property is empty. Using the BackgroundExt property you have unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the object's background. *For instance, let's say you need to display **more** colors on the object's background, or just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

The following screen shot shows a few possibilities you can have using the BackgroundExt property:



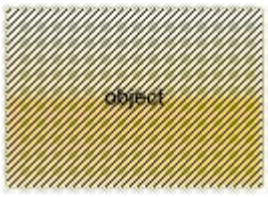
Complex samples:



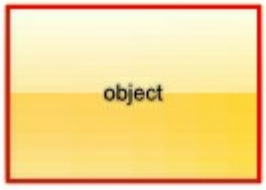


Easy samples:

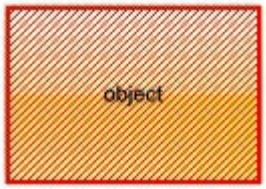
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



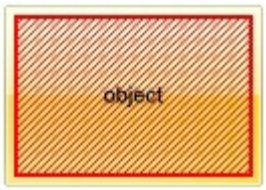
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



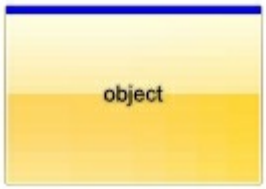
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



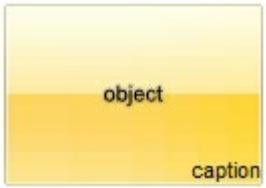
- "[[patterncolor=RGB(255,0,0)](none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



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

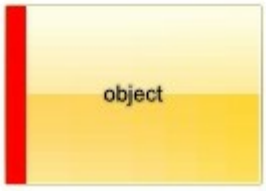


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

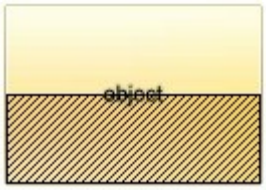


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

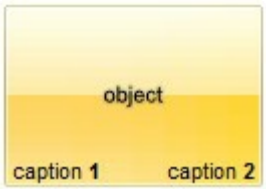
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



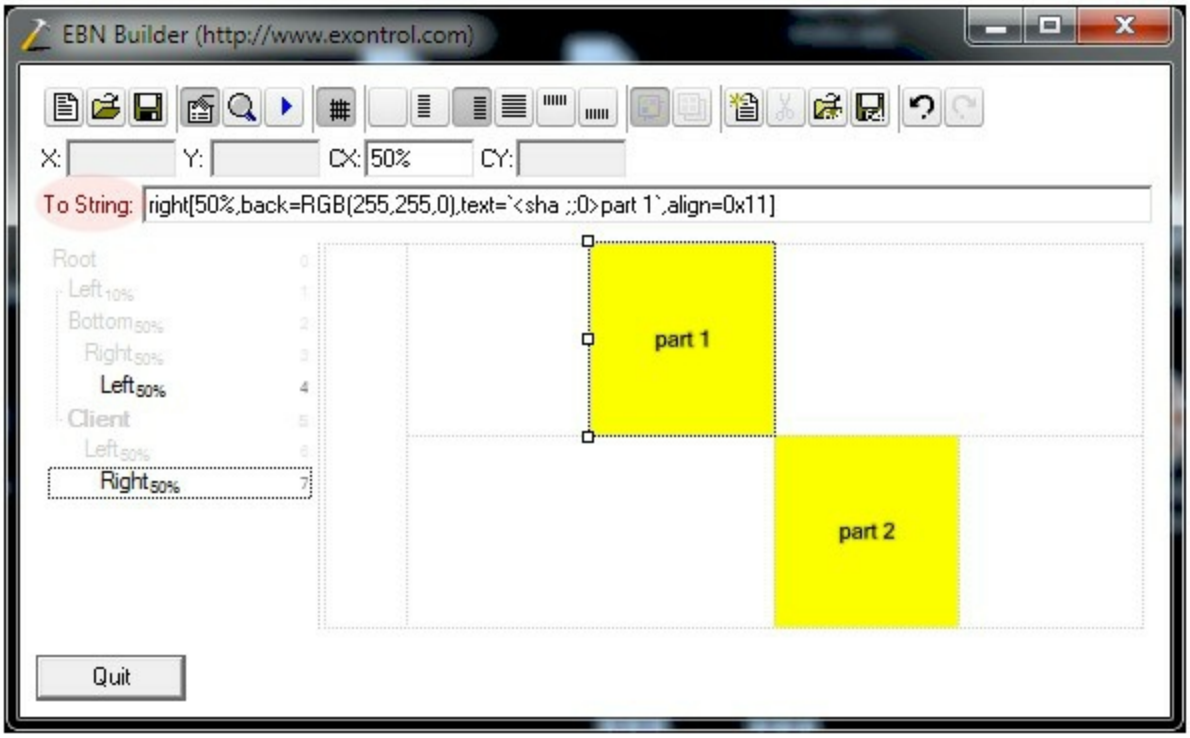
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



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



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



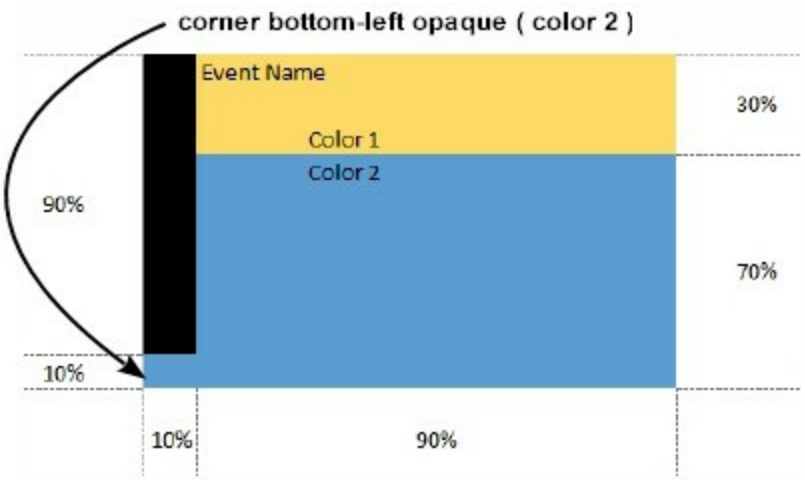
The **To String** field of the EBN Builder defines the **EBN String Format** that can be used on BackgroundExt property.

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

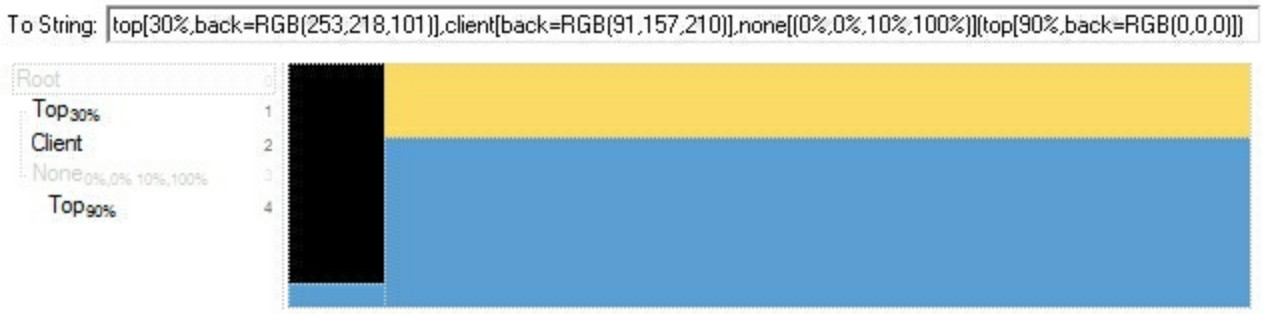
```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::=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

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

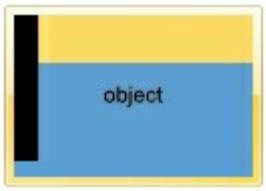
Now, lets say we have the following request to layout the colors on the objects:



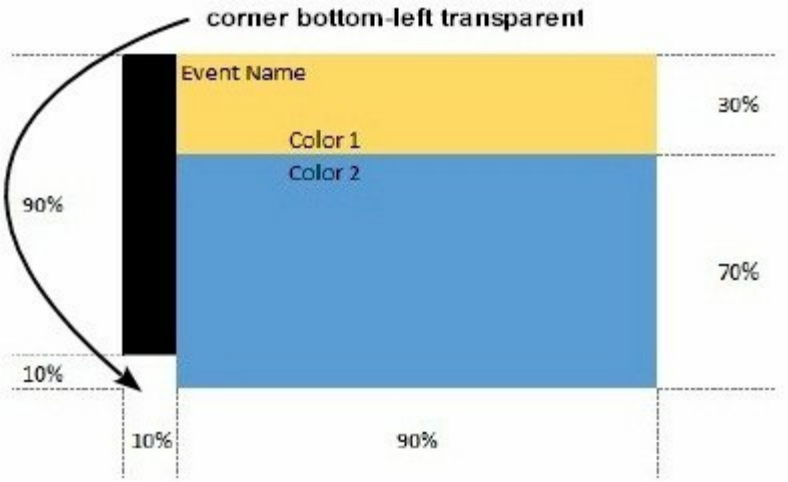
We define the BackgroundExt property such as "top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100%)](top[90%,back=RGB(0,0,0)])", and it looks as:



so, if we apply to our object we got:

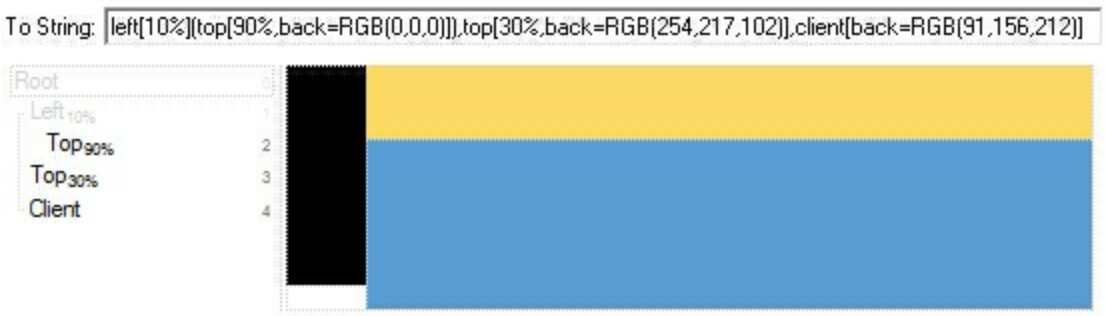


Now, lets say we have the following request to layout the colors on the objects:

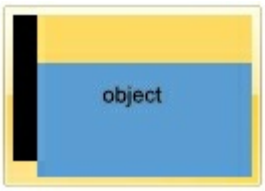


We define BackgroundExt property such as "left[10%]

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

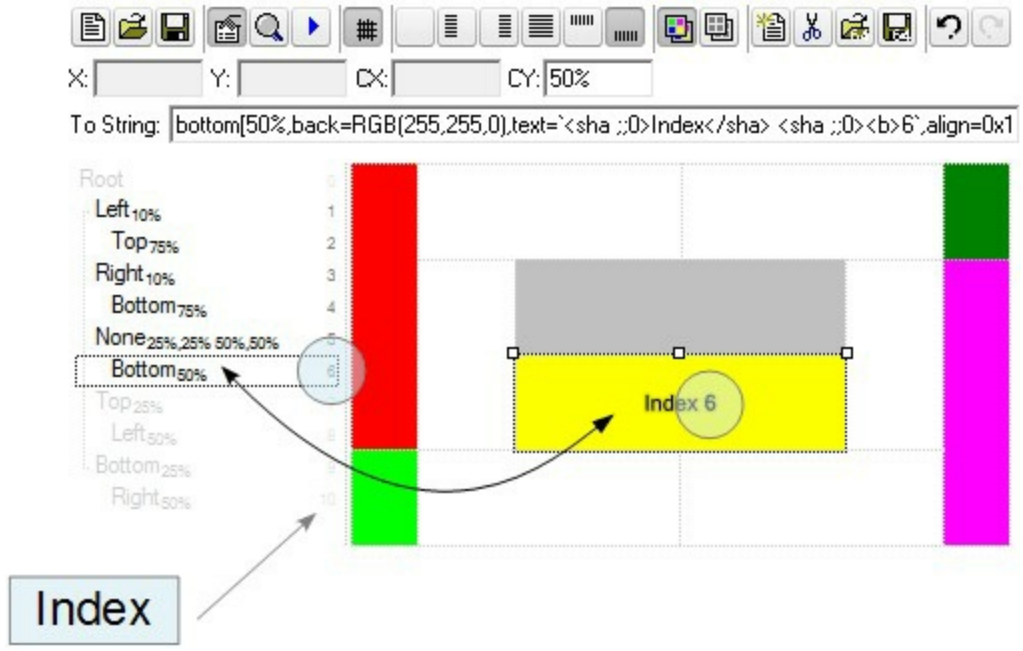


so, if we apply to our object we got:



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

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

Type	Description
Index as IndexExtEnum	A Long expression that defines the index of the part that composes the EBN to be accessed / changed. The following screen shot shows where you can find Index of the parts:  The screen shot shows that the EBN contains 11 elements, so in this case the Index starts at 0 (root element) and ends on 10.
Property as BackgroundExtPropertyEnum	A BackgroundExtPropertyEnum expression that specifies the property to be changed as explained bellow.
Variant	A Variant expression that defines the part's value. The Type of the expression depending on the Property parameter as explained bellow.

Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExtValue property has no effect if the BackgroundExt property is empty (by default). *The idea is as follows: first you need to decide the layout of the UI to put on the object's background, using the BackgroundExt*

property, and next (if required), you can change any property of any part of the background extension to a new value. In other words, let's say you have the same layout to be applied to some of your objects, so you specify the BackgroundExt to be the same for them, and next use the BackgroundExtValue property to change particular properties (like back-color, size, position, anchor) for different objects.

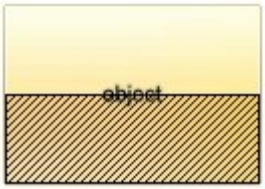
You can access/define/change the following UI properties of the element:

- **exBackColorExt**(1), Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000. (Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*
- **exClientExt**(2), Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect. Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent. (String/Numeric expression)*
- **exAnchorExt**(3), Specifies the object's alignment relative to its parent. *(Numeric expression)*
- **exTextExt**(4), Specifies the HTML text to be displayed on the object. *(String expression)*
- **exTextExtWordWrap**(5), Specifies that the object is wrapping the text. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *(Boolean expression)*
- **exTextExtAlignment**(6), Indicates the alignment of the text on the object. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag *(Numeric expression)*
- **exPatternExt**(7), Indicates the pattern to be shown on the object. The exPatternColorExt specifies the color to show the pattern. *(Numeric expression)*
- **exPatternColorExt**(8), Indicates the color to show the pattern on the object. The exPatternColorExt property has effect only if the exPatternExt property is not 0 (empty). The exFrameColorExt specifies the color to show the frame (the exPatternExt property includes the exFrame or exFrameThick flag). *(Color expression)*
- **exFrameColorExt**(9), Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. *(Color expression)*
- **exFrameThickExt**(11), Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property. *(Boolean expression)*
- **exUserDataExt**(12), Specifies an extra-data associated with the object. *(Variant*

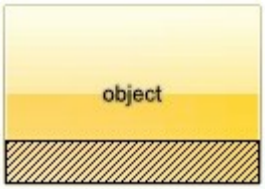
expression)

For instance, having the BackgroundExt on "bottom[50%,pattern=6,frame]"

we got:



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



In VB you should have the following syntax:

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

property Event.Bold as Boolean

Retrieves or sets a value that indicates whether the event should appear in bold.

Type	Description
Boolean	A boolean expression that indicates whether the date event is bolded.

Use the Events property to access the Events collection. The following sample shows how to bold the "tomorrow" date:

```
Calendar1.Events.Add(Date() + 1).Bold = True
```

property Event.Caption as String

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

Type	Description
String	A String expression that defines the HTML caption to be shown on the date.

By default, the Caption property is "**<%day%>**", which indicates that the day of the event is displayed on the date. Use the Caption property to display captions, icons, pictures to any date in the calendar panel. Use the <img> HTML tag to display icons loaded using the Use the Images method (which assign new icons to the control), or using the HTMLPicture property. For instance, the "*<sha><%day%><r><off -4><sha;;0><fgcolor FF0000>ev</sha></sha>*" displays the day and the "ev" word with different HTML formats. The BackgroundExt property provides unlimited options to show additional colors, text, icons, images, frames, patterns, ... to any event.

The following screen shot shows HTML captions on dates:

The Caption property supports the following HTML tags:

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

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

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

HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

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

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Event.Comment as String

Retrieves or sets a value that indicates the event's comment. (The comment shows up when the cursor is over the event).

Type	Description
String	A string expression that defines the event's comment. The Value supports HTML format like shown bellow.

If the Event object has associated a comment or a comment's description, a tooltip appears if the cursor is over the event's date. Use the ComentTitle to defines the title for the event's comment. Use the properties like: ToolTipDelay, ToolTipPopDelay, ToolTipFont and ToolTipWidth properties to specify the options for the tooltips. Use the CommentBackColor property to specify the color being used to mark the dates with comments. Use the Caption property to display a HTML caption to the event.

The Comment property supports the following HTML tags:

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

```
Private Sub Form_Load()
   With CalendarCombo1
      With .Events.Add(Date)
         .CommentTitle = "Just a title"
         .Comment = "This is a bit of text that should appear when the mouse is over the
date.<br><r><dotline><upline><b><fgcolor=0000FF>Exontrol ExCalendar</b>
</fgcolor>"
      End With
   End With
End Sub
```


9/16/2003
September 2003
October 2003
S M T W T F S
31 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
S M T W T F S
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
1
8
Just a title
This is a bit of text that should appear when the mouse is over the date.
Exontrol ExCalendar
Today

property Event.CommentTitle as String

Retrieves or sets a value that indicates the event's comment title.

Type	Description
String	A string expression that indicates the title for the for date's tooltip.

If the an Event object has associated a comment or a comment's description, a tooltip appears if the cursor is over the event's date. Use the ComentTitle property to defines the comment's description. Use the Comment property to define the cell's tooltip information.

The following sample shows how to attach a comment to "a day before yesterday" date:

```
Calendar1.Events.Add(Date - 2).CommentTitle = "A comment description"
Calendar1.Events.Add(Date - 2).Comment = "A comment"
```

property Event.Date as Date

Gets the event's date.

Type	Description
Date	A DATE expression that indicates the event's date

The property is read-only. To change the event date use the Add method of Events collection.

property Event.Disabled as Boolean

Retrieves or sets a value that indicates whether the event is disabled or enabled.

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

By default, the Disabled property is False. Use the Disabled property to disable a date. A disabled date looks grayed, and can't be selected. Use the FocusDate property to specify the date that has the focus. Use the Image property to assign an icon to a date. Use the ForeColor property to specify the date's foreground color whether the date is enabled. Use the BackColor property to specify the date's background color.

property Event.ForeColor as Color

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

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

Use the ForeColor and BackColor properties to set the foreground and background colors of the event.

property Event.Image as Long

Retrieves or sets the event's image.

Type	Description
Long	A long expression that indicates the index of the event's image into Images collection.

Use the Images method of the control to change the Images collection at runtime. The Image property is 1 based. If the index is not found into Images collection, no image is displayed on cell's date. If the date's image is not visible, use the AutoSize, FixedCellHeight and FixedCellWidth properties to increase the cell size. The cell size show be larger than (16,16) in order to display images. The Images method sets the control's handle image list. The ImageSize property defines the size (width/height) of the icons within the control's Images collection.

property Event.Italic as Boolean

Retrieves or sets a value that indicates whether the event should appear in italic.

Type	Description
Boolean	A boolean expression that indicates whether the event should appear in italic.

The following sample shows how to apply Italic font attribute to "today" date:

```
Calendar1.Events.Add(Date).Italic = True
```

property Event.Marker as Boolean

Retrieves or sets a value that indicates whether the event's marker is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the event's marker is visible or hidden.

By default, the Marker property is false. The event shows a frame around the date while Marker property is True. The Background(exMarkerColor) property Specifies the color or the visual appearance to apply on dates with Marker property set.

property Event.Repetitive as String

Returns or sets the expression to determine the repetitive event.

Type	Description
String	A String expression that defines the formula to determine the recurrence of the current event. The Repetitive property supports the **value** keyword and operators and expressions like defined bellow.

By default, the Repetitive property is "", which indicates that the event is not a repetitive event. The event is not repetitive if the Repetitive property is empty, blank or invalid.

The control supports two ways of representing a Repetitive/Recurrence event:

- **Value** format, when using the **value** keyword. For instance, "weekday(value) = 1", the event occurs every Monday
- **ICalendar** format, as described in RFC 5545. For instance, "FREQ=WEEKLY;BYDAY=MO", the event occurs every Monday

The FREQ property determines whether the Repetitive property uses the Value or ICalendar format. In other words, if the Repetitive property contains the FREQ keyword, the ICalendar format is using, else the Value format.

Here's a few samples of Repetitive expressions:

- "0", no occurrence
- "1", the event occurs every day
- "weekday(value) = 1", the event occurs every Monday
- "weekday(value) in (1,2) and month(value) = 6", the event occurs every Monday and Tuesday, on June only.
- "value in (#6/8/2012#,#6/11/2012#,#6/20/2012#)", the event occurs on 6/8/2012, 6/11/2012 and 6/20/2012
- "value >= #6/1/2012# and ((value - #6/1/2012#) mod 5 = 0)", the event starts on 6/1/2012, and shows up every 5 days
- "(value >= (0:=#6/1/2012#)) and ((value - =:0) mod (1:=5) = 0) and (value-=:0) < (3*=:1)", the event starts on 6/1/2012, occurs every 5 days, for 3 times. You can change 6/1/2012 with your date to indicates the starting date, changes 5 to indicate the n-occurrence and change 3 to indicate the m-times, so the event is shown every n-days for m-times.
- "not(month(value) in (3,4,5)) ? 0 : (floor(value)=(2:=floor(date(dateS('3/1/' + year(value)) + ((1:=(((255 - 11 * (year(value) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(value) + int(year(value) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7)))))", indicates the Easter- Sunday, so the event shows every year on Easter sunday.

The Repetitive property supports the **value** keyword which indicates the date being queried, and the following predefined operators and functions.

The supported binary arithmetic operators are:

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

The supported unary boolean operators are:

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

The supported binary boolean operators are:

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

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

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

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

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

  "expression ? true_part : false_part"

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

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

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

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

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

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

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

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

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

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

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

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

The syntax for *case()* operator is:

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

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

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

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

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

The supported conversion unary operators are:

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

  Here's few predefined types:

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

    - 7 - date
    - 8 - string
    - 9 - object
    - 10 - error
    - 11 - boolean
    - 12 - variant
    - 13 - any
    - 14 - decimal
    - 16 - char
    - 17 - byte
    - 18 - unsigned short
    - 19 - unsigned long
    - 20 - long on 64 bits
    - 21 - unsigned long on 64 bites
- **str** (unary operator) converts the expression to a string
- **dbl** (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date, based on your regional settings. The **date(``)** returns now ( date + time ), and **int(date(``))** gets the today date (with no time including)
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS.

Other known operators for numbers are:

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

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

    - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.

- *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
- Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

Other known operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string
- **lower** (unary operator) returns a string expression in lowercase letters
- **upper** (unary operator) returns a string expression in uppercase letters
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names
- **ltrim** (unary operator) removes spaces on the left side of a string
- **rtrim** (unary operator) removes spaces on the right side of a string
- **trim** (unary operator) removes spaces on both sides of a string
- **startwith** (binary operator) specifies whether a string starts with specified string
- **endwith** (binary operator) specifies whether a string ends with specified string
- **contains** (binary operator) specifies whether a string contains another specified string
- **left** (binary operator) retrieves the left part of the string
- **right** (binary operator) retrieves the right part of the string
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on)

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

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- **year** (unary operator) retrieves the year of the date (100,...,9999)
- **month** (unary operator) retrieves the month of the date (1, 2,...,12)
- **day** (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday)
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59)

The BNF syntax for ICalendar format is:

```
recur = recur-rule-part *( ";" recur-rule-part )
   ;
   ; The rule parts are not ordered in any
   ; particular sequence.
   ;
   ; The FREQ rule part is REQUIRED,
   ; but MUST NOT occur more than once.
```

```
   ;
   ; The UNTIL or COUNT rule parts are OPTIONAL,
   ; but they MUST NOT occur in the same 'recur'.
   ;
   ; The other rule parts are OPTIONAL,
   ; but MUST NOT occur more than once.

recur-rule-part = ( "FREQ" "=" freq )
             / ( "UNTIL" "=" enddate )
             / ( "COUNT" "=" 1*DIGIT )
             / ( "INTERVAL" "=" 1*DIGIT )
             / ( "BYSECOND" "=" byseclist )
             / ( "BYMINUTE" "=" byminlist )
             / ( "BYHOUR" "=" byhrlist )
             / ( "BYDAY" "=" bywdaylist )
             / ( "BYMONTHDAY" "=" bymodaylist )
             / ( "BYYEARDAY" "=" byyrdaylist )
             / ( "BYWEEKNO" "=" bywknolist )
             / ( "BYMONTH" "=" bymolist )
             / ( "BYSETPOS" "=" bysplist )
             / ( "WKST" "=" weekday )

freq       = "SECONDLY" / "MINUTELY" / "HOURLY" / "DAILY"
          / "WEEKLY" / "MONTHLY" / "YEARLY"
enddate    = date / date-time
byseclist  = ( seconds *("," seconds) )
seconds    = 1*2DIGIT      ;0 to 60
byminlist  = ( minutes *("," minutes) )
minutes    = 1*2DIGIT      ;0 to 59
byhrlist   = ( hour *("," hour) )
hour       = 1*2DIGIT      ;0 to 23
bywdaylist  = ( weekdaynum *("," weekdaynum) )
weekdaynum  = [[plus / minus] ordwk] weekday
plus       = "+"
minus      = "-"
ordwk      = 1*2DIGIT      ;1 to 53
weekday    = "SU" / "MO" / "TU" / "WE" / "TH" / "FR" / "SA" ;Corresponding to SUNDAY,
```

```
MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, and SATURDAY days of the week.
bymodaylist = ( monthdaynum *("," monthdaynum) )
monthdaynum = [plus / minus] ordmoday
ordmoday    = 1*2DIGIT      ;1 to 31
byyrdaylist = ( yeardaynum *("," yeardaynum) )
yeardaynum  = [plus / minus] ordyrday
ordyrday    = 1*3DIGIT      ;1 to 366
bywknolist  = ( weeknum *("," weeknum) )
weeknum     = [plus / minus] ordwk
bymolist    = ( monthnum *("," monthnum) )
monthnum    = 1*2DIGIT      ;1 to 12
bysplist    = ( setposday *("," setposday) )
setposday   = yeardaynum
```

This value type is a structured value consisting of a list of one or more recurrence grammar parts. Each rule part is defined by a NAME=VALUE pair. The rule parts are separated from each other by the SEMICOLON character. The rule parts are not ordered in any particular sequence. Individual rule parts MUST only be specified once. Compliant applications MUST accept rule parts ordered in any sequence, but to ensure backward compatibility with applications that pre-date this revision of iCalendar the FREQ rule part MUST be the first rule part specified in a RECUR value.

The FREQ rule part identifies the type of recurrence rule. This rule part MUST be specified in the recurrence rule. Valid values include SECONDLY, to specify repeating events based on an interval of a second or more; MINUTELY, to specify repeating events based on an interval of a minute or more; HOURLY, to specify repeating events based on an interval of an hour or more; DAILY, to specify repeating events based on an interval of a day or more; WEEKLY, to specify repeating events based on an interval of a week or more; MONTHLY, to specify repeating events based on an interval of a month or more; and YEARLY, to specify repeating events based on an interval of a year or more.

The INTERVAL rule part contains a positive integer representing at which intervals the recurrence rule repeats. The default value is "1", meaning every second for a SECONDLY rule, every minute for a MINUTELY rule, every hour for an HOURLY rule, every day for a DAILY rule, every week for a WEEKLY rule, every month for a MONTHLY rule, and every year for a YEARLY rule. For example, within a DAILY rule, a value of "8" means every eight days.

The UNTIL rule part defines a DATE or DATE-TIME value that bounds the recurrence rule in an inclusive manner. If the value specified by UNTIL is synchronized with the specified recurrence, this DATE or DATE-TIME becomes the last instance of the recurrence. The

value of the UNTIL rule part MUST have the same value type as the "DTSTART" property. Furthermore, if the "DTSTART" property is specified as a date with local time, then the UNTIL rule part MUST also be specified as a date with local time. If the "DTSTART" property is specified as a date with UTC time or a date with local time and time zone reference, then the UNTIL rule part MUST be specified as a date with UTC time. In the case of the "STANDARD" and "DAYLIGHT" sub-components the UNTIL rule part MUST always be specified as a date with UTC time. If specified as a DATE-TIME value, then it MUST be specified in a UTC time format. If not present, and the COUNT rule part is also not present, the "RRULE" is considered to repeat forever.

The COUNT rule part defines the number of occurrences at which to range-bound the recurrence. The "DTSTART" property value always counts as the first occurrence.

The BYSECOND rule part specifies a COMMA-separated list of seconds within a minute. Valid values are 0 to 60. The BYMINUTE rule part specifies a COMMA-separated list of minutes within an hour. Valid values are 0 to 59. The BYHOUR rule part specifies a COMMA- separated list of hours of the day. Valid values are 0 to 23. The BYSECOND, BYMINUTE and BYHOUR rule parts MUST NOT be specified when the associated "DTSTART" property has a DATE value type. These rule parts MUST be ignored in RECUR value that violate the above requirement (e.g., generated by applications that pre-date this revision of iCalendar).

The BYDAY rule part specifies a COMMA-separated list of days of the week; SU indicates Sunday; MO indicates Monday; TU indicates Tuesday; WE indicates Wednesday; TH indicates Thursday; FR indicates Friday; and SA indicates Saturday. Each BYDAY value can also be preceded by a positive (+n) or negative (-n) integer. If present, this indicates the nth occurrence of a specific day within the MONTHLY or YEARLY "RRULE".
For example, within a MONTHLY rule, +1MO (or simply 1MO) represents the first Monday within the month, whereas -1MO represents the last Monday of the month. The numeric value in a BYDAY rule part with the FREQ rule part set to YEARLY corresponds to an offset within the month when the BYMONTH rule part is present, and corresponds to an offset within the year when the BYWEEKNO or BYMONTH rule parts are present. If an integer modifier is not present, it means all days of this type within the specified frequency. For example, within a MONTHLY rule, MO represents all Mondays within the month. The BYDAY rule part MUST NOT be specified with a numeric value when the FREQ rule part is not set to MONTHLY or YEARLY. Furthermore, the BYDAY rule part MUST NOT be specified with a numeric value with the FREQ rule part set to YEARLY when the BYWEEKNO rule part is specified.

The BYMONTHDAY rule part specifies a COMMA-separated list of days of the month. Valid values are 1 to 31 or -31 to -1. For example, -10 represents the tenth to the last day of the month. The BYMONTHDAY rule part MUST NOT be specified when the FREQ rule part is set to WEEKLY.

The BYYEARDAY rule part specifies a COMMA-separated list of days of the year. Valid values are 1 to 366 or -366 to -1. For example, -1 represents the last day of the year (December 31st) and -306 represents the 306th to the last day of the year (March 1st). The BYYEARDAY rule part MUST NOT be specified when the FREQ rule part is set to DAILY, WEEKLY, or MONTHLY.

The BYWEEKNO rule part specifies a COMMA-separated list of ordinals specifying weeks of the year. Valid values are 1 to 53 or -53 to -1. This corresponds to weeks according to week numbering as defined in [ISO.8601.2004]. A week is defined as a seven day period, starting on the day of the week defined to be the week start (see WKST). Week number one of the calendar year is the first week that contains at least four (4) days in that calendar year. This rule part MUST NOT be used when the FREQ rule part is set to anything other than YEARLY. For example, 3 represents the third week of the year.

Note: Assuming a Monday week start, week 53 can only occur when Thursday is January 1 or if it is a leap year and Wednesday is January 1.

The BYMONTH rule part specifies a COMMA-separated list of months of the year. Valid values are 1 to 12.

The WKST rule part specifies the day on which the workweek starts. Valid values are MO, TU, WE, TH, FR, SA, and SU. This is significant when a WEEKLY "RRULE" has an interval greater than 1, and a BYDAY rule part is specified. This is also significant when in a YEARLY "RRULE" when a BYWEEKNO rule part is specified. The default value is MO.

The BYSETPOS rule part specifies a COMMA-separated list of values that corresponds to the nth occurrence within the set of recurrence instances specified by the rule. BYSETPOS operates on a set of recurrence instances in one interval of the recurrence rule. For example, in a WEEKLY rule, the interval would be one week A set of recurrence instances starts at the beginning of the interval defined by the FREQ rule part. Valid values are 1 to 366 or -366 to -1. It MUST only be used in conjunction with another BYxxx rule part. For example "the last work day of the month" could be represented as:

FREQ=MONTHLY;BYDAY=MO,TU,WE,TH,FR;BYSETPOS=-1

Each BYSETPOS value can include a positive (+n) or negative (-n) integer. If present, this indicates the nth occurrence of the specific occurrence within the set of occurrences specified by the rule.

Recurrence rules may generate recurrence instances with an invalid date (e.g., February 30) or nonexistent local time (e.g., 1:30 AM on a day where the local time is moved forward by an hour at 1:00 AM). Such recurrence instances MUST be ignored and MUST NOT be counted as part of the recurrence set. Information, not contained in the rule, necessary to determine the various recurrence instance start time and dates are derived from the Start

Time ("DTSTART") component attribute. For example, "FREQ=YEARLY;BYMONTH=1" doesn't specify a specific day within the month or a time. This information would be the same as what is specified for "DTSTART".

property Event.StrikeOut as Boolean

Retrieves or sets a value that indicates whether the event should appear in strikeout.

Type	Description
Boolean	A boolean expression that indicates whether the event should appear in strikeout.

The following sample shows how to apply Strikeout font attribute to "today" date:

```
Calendar1.Events.Add(Date).StrikeOut = True
```

property Event.Underline as Boolean

Retrieves or sets a value that indicates whether the event appears as underlined.

Type	Description
Boolean	A boolean expression that indicates whether the event appears as underlined.

The following sample shows how to apply Underline font attribute to "today" date:

```
Calendar1.Events.Add(Date).Underline = True
```

property Event.UserData as Variant

Retrieves or sets the event's user associated extra data.

Type	Description
Variant	A VARIANT value associated to the event

Use the UserData property to associate any extra data to the event.

Events object

The Events collection contains Event objects. Each Event object has associated a date. An Event object contains graphical information about the date, like: colors, markers, tooltips, images, and so on. Use the Add method to add new elements to the collection. Use the Serialize property to save or load the collection to/from a string

Name	Description
Add	Adds a new date event, and retrieves the newly created date event.
Clear	Clears the events collection.
Count	Retrieves the count of elements within collection.
Item	Gets the Event object based on its index or date.
Remove	Removes a date event based on its index or a date.
Serialize	Serializes the collection of events to a string, for later use.

method Events.Add (Event as Date)

Adds a new date event, and retrieves the newly created date event.

Type	Description
Event as Date	A DATE expression that indicates the event's date

Return	Description
Event	An Event object that holds graphical information about a date.

Use Add property to adds a new element to the collection. Each Event object has associated a date. If the Events collection contains already an event associated to the date, the Add method retrieves the reference to the associated event. If the date is not associated to none of the Event objects, a new Event object is added to the collection. The Date property is initialized with the event parameter.

The following sample shows how to bold "today" date:

```
Calendar1.Events.Add(Date).Bold = True
```

method Events.Clear ()

Clears the events collection.

Type	Description

Use Remove method to remove an element from the collection. The Clear method does refresh the control's client area.

property Events.Count as Long

Retrieves the count of elements within collection.

Type	Description
Long	A long expression that indicates the number of elements into collection.

The Count property counts the Event objects in the Events collection.

property Events.Item (Index as Variant) as Event

Gets the Event object based on its index or date.

Type	Description
Index as Variant	A long expression that indicates the index of Event into Events collection, or a DATE expression.
Event	An Event object requested.

Use the Item property to enumerate the elements into collection. You can use Add property as well as Item property, when you have a DATE expression. The Add method wont add two Event object with the same Date.

method Events.Remove (Index as Variant)

Removes a date event based on its index or a date.

Type	Description
Index as Variant	A long expression that indicates the index of Event into Events collection, or a DATE expression.

Use Clear method to clear the Events collection.

property Events.Serialize as String

Serializes the collection of events to a string, for later use.

Type	Description
String	A string expression that indicates the encoded collection.

Save and load the collection to/from a string. Use the Serialize property to save the collection to a string. use the Serialize property to load the Events collection, that was previously saved by Serialize property.

CalendarCombo events

The CalendarCombo object supports the following properties and methods:

Name	Description
Click	Occurs when the user presses and then releases the left mouse button over the control.
DateChanged	Fired when the browsed date is changed.
DateChanging	Occurs when the user is about to change the browsed date.
DblClick	Occurs when the user dblclk the left mouse button over an object.
Event	Notifies the application once the control fires an event.
FocusIndexChanged	Occurs when the index of the focused element is changed.
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.
RClick	Fired when the user releases the right mouse button.
Select	Notifies the application once the user hits ENTER key or selects a date using the mouse.
SelectionChanged	Fired when the selection is changed.
UIChange	Occurs when the control's user interface is changed.

event Click ()

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

Type	Description

The Click event is fired when the user releases the left mouse button over the control. Use the DblClick event to notify your application when the user double clicks the control's label. Use the RClick event to notify your application when the user right clicks the control's label.

event DateChanged ()

Fired when the browsed date is changed.

Type	Description

The control fires the DateChanged event when a new date is browsed, in the drop down portion of the control. Use the Date property to change the browsed date. The DateChanged event is fired when the drop down portion of the control browses a new date. The DateChaniging event occurs when the user is about to change the browsed date by clicking any of the arrows in the calendar window. The Value property is identical with the SelDate property. Please make sure that the SelectionChanged event property is fired when the control's value is changed. The SelDate property indicates the selected date. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date.

The following sample displays the browsed date:

```
Private Sub CalendarCombo1_DateChanged()
   Debug.Print FormatDateTime(CalendarCombo1.Date)
End Sub
```

event DateChanging (HeaderArrow as HeaderArrowEnum, Cancel as Variant)

Occurs when the user is about to change the browsed date.

Type	Description
HeaderArrow as HeaderArrowEnum	A HeaderArrowEnum expression that indicates the arrow being clicked.
Cancel as Variant	A Boolean expression that specifies whether the default operation is canceled or not.

Use the DataChanging property event to increase or decrease the browsed date as user clicks the arrows in the header of the CalendarCombo. The Date property specifies the browsed date. The DateChanged event notifies your application that the browsed date is changed. By default, the Cancel parameter is False, so the default operation is executed when the user clicks a button in the header of the CalendarCombo. The SelDate property indicates the date being selected.

The following VB sample changes the month to prev or next when user clicks the LEFT or RIGHT buttons, instead prev / next Year (which is by default).

```
Private Sub CalendarCombo1_DateChanging(ByVal HeaderArrow As
EXCalendarComboLibCtl.HeaderArrowEnum, Cancel As Variant)
   Cancel = True
   With CalendarCombo1
      If (HeaderArrow = exPrevYear) Then
         .Date = DateAdd("m", -1, .Date)
      Else
         If (HeaderArrow = exNextYear) Then
            .Date = DateAdd("m", 1, .Date)
         End If
      End If
   End With
End Sub
```

Previously the **ShowMonthSelector property is set on False**, so only the LEFT and RIGHT buttons are displayed. By default, the LEFT and RIGHT arrows changes the year, while the UP and DOWN arrows changes the month. The sample hides the UP and DOWN buttons, and let only the LEFT and RIGHT buttons and the code changes the current date to prev or next month.

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.

Use the DblClick event to notify your application when the user double clicks the control's label. The Click event is fired when the user releases the left mouse button over the control. Use the RClick event to notify your application when the user right clicks the control's label. Use the IndexFromPoint property to determine the index of the date's element from the point. Use the Caption property to retrieve the caption of the date's element.

The following VB sample prints the date's element being double clicked:

```
Private Sub CalendarCombo1_DblClick(Shift As Integer, X As Single, Y As Single)
  With CalendarCombo1
    h = .IndexFromPoint(-1, -1)
    Debug.Print .Caption(h)
  End With
End Sub
```

The following VB.NET sample prints the date's element being double clicked:

```
Private Sub AxCalendarCombo1_DblClick(ByVal sender As Object, ByVal e As
AxEXCALENDARLib._ICalendarComboEvents_DblClickEvent) Handles
AxCalendarCombo1.DblClick
  With AxCalendarCombo1
    System.Diagnostics.Debug.Print(.get_Caption(.get_IndexFromPoint(e.x, e.y)))
  End With
End Sub
```

The following C# sample prints the date's element being double clicked:

```
private void axCalendarCombo1_DblClick(object sender,
AxEXCALENDARLib._ICalendarComboEvents_DblClickEvent e)
{

System.Diagnostics.Debug.Print(axCalendarCombo1.get_Caption(axCalendarCombo1.get_I
 e.y)));
}
```

The following C++ sample prints the date's element being double clicked:

```
void OnDblClickCalendarcombo1(short Shift, long X, long Y)
{
    long i = m_calendarcombo.GetIndexFromPoint( X, Y );
    OutputDebugString( m_calendarcombo.GetCaption( COleVariant( i ) ) );
}
```

The following VFP sample prints the date's element being double clicked:

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

with thisform.CalendarCombo1
    ?.Caption(.IndexFromPoint(x,y))
endwith
```

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. Each internal event of the control has an unique identifier. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties). The EventParam(-1) retrieves the number of parameters of fired event

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

Click here to watch a movie on how you can use the eXHelper to get information about the fired events using the Event handler. The Event notification is sent any time the control fires a specified event.

This is useful for X++, which does not support event with parameters passed by reference. Also, this could be useful for C++ Builder or Delphi, which does not handle properly the events with parameters of VARIANT type.

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

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

Let's assume that we need to handle the BarParentChange event to change the _Cancel parameter from false to true, which fires the "Error executing code: FormActiveXControl (data source), method onEvent_BarParentChange called with invalid parameters." We need to know the identifier of the BarParentChange event (each event has an unique identifier and it is static, defined in the control's type library). If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print excalendarcombo1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

event FocusIndexChanged ()

Occurs when the index of the focused element is changed.

Type	Description

The FocusIndexChanged property notifies your application when the FocusIndex property is changed. The FocusIndex property specifies the index of the element being focused in the control's label. The FocusIndexChanged property is fired if the FocusIndex property is changed at runtime, or changing by code.

event KeyDown (KeyCode as Integer, Shift as Integer)

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

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

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

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

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

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

event KeyUp (KeyCode as Integer, Shift as Integer)

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

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

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

event RClick ()

Fired when the user releases the right mouse button.

Type	Description

Use the RClick event to notify your application when the user right clicks the control's label. The Click event is fired when the user releases the left mouse button over the control. Use the DblClick event to notify your application when the user double clicks the control's label.

event Select ()

Notifies the application once the user hits ENTER key or selects a date using the mouse.

Type	Description

event SelectionChanged ()

Fired when the selection was changed.

Type	Description

Use the SelectionChanged event to notify your application when the user changes the selected date. Use the SelDate property to get or set the selected date. The Date property specifies the date being browsed in the drop down portion of the control. Use the MaskOnEmpty property to specify the caption being displayed in the control's label when the SelDate property is empty.

The following sample prints the selected date:

```
Private Sub CalendarCombo1_SelectionChanged()
   Debug.Print FormatDateTime(CalendarCombo1.SelDate())
End Sub
```

event UIChange (Change as UIChangeEnum)

Occurs when the control's user interface is changed.

Type	Description
Change as UIChangeEnum	An UIChangeEnum expression that specifies the state and the part of the control's UI to be changed or being changed.

Use the UIChange property to notify your application once the user clicks the drop down's check box, or when the user clicks the control's spin buttons.

Calendar events

The ExCalendar object supports the following properties and methods:

Name	Description
Click	Occurs when the user presses and then releases the left mouse button over the calendar control.
DateChanged	Fired when the browsed date is changed.
DateChanging	Occurs when the user is about to change the browsed date.
DblClick	Occurs when the user dblclk the left mouse button over an object.
Event	Notifies the application once the control fires an event.
FocusChanged	Fired when the focused date is changed.
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.
RClick	Fired when right mouse button is clicked
SelectionChanged	Fired when the selection is changed.

event Click ()

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

Type	Description

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

Fired when the browsed date is changed.

Type	Description

Use the DateChanged to notify your application when the browsed date was changed. The DateChanged event is fired if the Date property is changed. The DataChanging property notifies your application that the user is about to change the browsed date, by clicking any of arrows in the header of the calendar window. Use the SelectionChanged event to notify your application when the selection was changed. The SelectionChanged event is fired when user changes the SelDate property. Use the FirstVisibleDate property to get the first visible date. Use the LastVisibleDate property to get the last visible date.

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

C#

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

VB

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

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

C#

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

C++

```
void OnDateChanged()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event DateChanged()
end event DateChanged
```

VB.NET

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

VB6

```
Private Sub DateChanged()
End Sub
```

VBA

```
Private Sub DateChanged()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnDateChanged(oCalendar)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComDateChanged
    Forward Send OnComDateChanged
```

```
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_DateChanged()
{
}
```

XBasic

```
function DateChanged as v ()
end function
```

dBASE

```
function nativeObject_DateChanged()
return
```

The following sample displays the browsed date:

```
Private Sub Calendar1_DateChanged()
    Debug.Print FormatDateTime(Calendar1.Date)
End Sub
```

event DateChanging (HeaderArrow as HeaderArrowEnum, Cancel as Variant)

Occurs when the user is about to change the browsed date.

Type	Description
HeaderArrow as HeaderArrowEnum	A HeaderArrowEnum expression that indicates the arrow being clicked.
Cancel as Variant	A Boolean expression that specifies whether the default operation is canceled or not.

Use the DataChanging property event to increase or decrease the browsed date as user clicks the arrows in the header of the calendar. The Date property specifies the browsed date. The DateChanged event notifies your application that the browsed date is changed. By default, the Cancel parameter is False, so the default operation is executed when the user clicks a button in the header of the calendar. The SelDate property indicates the date being selected.

The following VB sample changes the month to prev or next when user clicks the LEFT or RIGHT buttons, instead prev / next Year (which is by default).

```
Private Sub Calendar1_DateChanging(ByVal HeaderArrow As
EXCALENDARLibCtl.HeaderArrowEnum, Cancel As Variant)
   Cancel = True
   With Calendar1
      If (HeaderArrow = exPrevYear) Then
         .Date = DateAdd("m", -1, .Date)
      Else
         If (HeaderArrow = exNextYear) Then
            .Date = DateAdd("m", 1, .Date)
         End If
      End If
   End With
End Sub
```

Previously the **ShowMonthSelector property is set on False**, so only the LEFT and RIGHT buttons are displayed. By default, the LEFT and RIGHT arrows changes the year, while the UP and DOWN arrows changes the month. The sample hides the UP and DOWN buttons, and let only the LEFT and RIGHT buttons and the code changes the current date to prev or next month.

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

C#

```
private void DateChanging(object
sender,exontrol.EXCALENDARLib.HeaderArrowEnum HeaderArrow,ref object
Cancel)
{
}
```

VB

```
Private Sub DateChanging(ByVal sender As System.Object,ByVal HeaderArrow As
exontrol.EXCALENDARLib.HeaderArrowEnum,ByRef Cancel As Object) Handles
DateChanging
End Sub
```

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

C#

```
private void DateChanging(object sender,
AxEXCALENDARLib._ICalendarComboEvents_DateChangingEvent e)
{
}
```

C++

```
void OnDateChanging(long HeaderArrow,VARIANT FAR* Cancel)
{
}
```

C++ Builder

```
void __fastcall DateChanging(TObject
*Sender,Excalendarlib_tlb::HeaderArrowEnum HeaderArrow,Variant * Cancel)
{
}
```

Delphi

```
procedure DateChanging(ASender: TObject; HeaderArrow : HeaderArrowEnum;var
Cancel : OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DateChanging(sender: System.Object; e:
AxEXCALENDARLib._ICalendarComboEvents_DateChangingEvent);
begin
end;
```

Powe…
```
begin event DateChanging(long HeaderArrow,any Cancel)
end event DateChanging
```

VB.NET
```
Private Sub DateChanging(ByVal sender As System.Object, ByVal e As
AxEXCALENDARLib._ICalendarComboEvents_DateChangingEvent) Handles
DateChanging
End Sub
```

VB6
```
Private Sub DateChanging(ByVal HeaderArrow As
EXCALENDARLibCtl.HeaderArrowEnum,Cancel As Variant)
End Sub
```

VBA
```
Private Sub DateChanging(ByVal HeaderArrow As Long,Cancel As Variant)
End Sub
```

VFP
```
LPARAMETERS HeaderArrow,Cancel
```

Xbas…
```
PROCEDURE OnDateChanging(oCalendar,HeaderArrow,Cancel)
RETURN
```

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

Java…
```
<SCRIPT EVENT="DateChanging(HeaderArrow,Cancel)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function DateChanging(HeaderArrow,Cancel)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComDateChanging OLEHeaderArrowEnum llHeaderArrow Variant
llCancel
    Forward Send OnComDateChanging llHeaderArrow llCancel
End_Procedure
```

Visual Objects
```
METHOD OCX_DateChanging(HeaderArrow,Cancel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DateChanging(int _HeaderArrow,COMVariant /*variant*/ _Cancel)
{
}
```

XBasic

```
function DateChanging as v (HeaderArrow as
OLE::Exontrol.Calendar.1::HeaderArrowEnum,Cancel as A)
end function
```

dBASE

```
function nativeObject_DateChanging(HeaderArrow,Cancel)
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 notifies your application that user double clicked the control's client area. Use the DateFromPoint property to retrieve the date from cursor.

The following sample prints the date from the cursor:

```
Private Sub Calendar1_DblClick(Shift As Integer, X As Single, Y As Single)
  With Calendar1
    Dim d As Date
    d = .DateFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    If Not (d = 0) Then
      Debug.Print FormatDateTime(d)
    End If
  End With
End Sub
```

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,
AxEXCALENDARLib._ICalendarComboEvents_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:
AxEXCALENDARLib._ICalendarComboEvents_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
AxEXCALENDARLib._ICalendarComboEvents_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(oCalendar,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.Calendar.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Calendar.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. Each internal event of the control has an unique identifier. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties). The EventParam(-1) retrieves the number of parameters of fired event

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

Click here to watch a movie on how you can use the eXHelper to get information about the fired events using the Event handler. The Event notification is sent any time the control fires a specified event.

This is useful for X++, which does not support event with parameters passed by reference. Also, this could be useful for C++ Builder or Delphi, which does not handle properly the events with parameters of VARIANT type.

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

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

Let's assume that we need to handle the BarParentChange event to change the _Cancel parameter from false to true, which fires the "Error executing code: FormActiveXControl (data source), method onEvent_BarParentChange called with invalid parameters." We need to know the identifier of the BarParentChange event (each event has an unique identifier and it is static, defined in the control's type library). If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print excalendar1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

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,
AxEXCALENDARLib._ICalendarComboEvents_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:
AxEXCALENDARLib._ICalendarComboEvents_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
AxEXCALENDARLib._ICalendarComboEvents_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(oCalendar,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 FocusChanged ()

Fired when the focused date is changed.

Type	Description

The FocusChanged event notifies your application that a new date is focused. use the FocusDate property to retrieve or sets the date that has the focus. Use the SelDate property to select a date. Use the Disabled property to disable a date. The SelectionChanged event is fired when user changes the selection. The control fires the DateChanged event when a new date is browsed.

The following VB sample prints the focused date when user changes selects a new date:

```
Private Sub Calendar1_FocusChanged()
   Debug.Print Calendar1.FocusDate
End Sub
```

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

C#

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

VB

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

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

C#

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

C++

```
void OnFocusChanged()
{
}
```

C++ Builder

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

Delphi
```
procedure FocusChanged(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)
```
procedure FocusChanged(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…
```
begin event FocusChanged()
end event FocusChanged
```

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

VB6
```
Private Sub FocusChanged()
End Sub
```

VBA
```
Private Sub FocusChanged()
End Sub
```

VFP
```
LPARAMETERS nop

```

Xbas…
```
PROCEDURE OnFocusChanged(oCalendar)
RETURN
```

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

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

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function FocusChanged()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComFocusChanged
    Forward Send OnComFocusChanged
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_FocusChanged()
{
}
```

XBasic

```
function FocusChanged as v ()
end function
```

dBASE

```
function nativeObject_FocusChanged()
return
```

event KeyDown (KeyCode as Integer, Shift as Integer)

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

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

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

```
ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
```

In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

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

C#

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

VB

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

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

C#

```
private void KeyDownEvent(object sender,
AxEXCALENDARLib._ICalendarComboEvents_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:
AxEXCALENDARLib._ICalendarComboEvents_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
AxEXCALENDARLib._ICalendarComboEvents_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(oCalendar,KeyCode,Shift)
RETURN
```

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

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

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

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

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

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

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

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

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

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

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

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

C#

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

VB

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

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

C#

```
private void KeyPressEvent(object sender,
AxEXCALENDARLib._ICalendarComboEvents_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:
AxEXCALENDARLib._ICalendarComboEvents_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
AxEXCALENDARLib._ICalendarComboEvents_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(oCalendar,KeyAscii)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComKeyPress Short llKeyAscii
    Forward Send OnComKeyPress llKeyAscii
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii)
{
}
```

XBasic

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

dBASE

```
function nativeObject_KeyPress(KeyAscii)
return
```

event KeyUp (KeyCode as Integer, Shift as Integer)

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

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

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

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

C#

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

VB

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

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

C#

```
private void KeyUpEvent(object sender,
AxEXCALENDARLib._ICalendarComboEvents_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:
AxEXCALENDARLib._ICalendarComboEvents_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
AxEXCALENDARLib._ICalendarComboEvents_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(oCalendar,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 X location of the mouse pointer. The X value is always expressed in container coordinates.

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

The following VB sample prints the date from the cursor:

```
Private Sub Calendar1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   With Calendar1
      Dim d As Date
      d = .DateFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (d = 0) Then
         Debug.Print FormatDateTime(d)
      End If
   End With
End Sub
```

The following Access sample displays a message when user clicks a date:

```
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
```

```
Private Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal hDC As Long) As Long
Private Declare Function GetDeviceCaps Lib "gdi32" (ByVal hDC As Long, ByVal nIndex As Long) As Long
Private Const LOGPIXELSX = 88
Private Const LOGPIXELSY = 90

' Converts twips corrdinates to pixels coordinates
Private Sub Twips2Pixels(X As Long, Y As Long)
   Dim hDC As Long
   hDC = GetDC(0)
   X = X / 1440 * GetDeviceCaps(hDC, LOGPIXELSX)
   Y = Y / 1440 * GetDeviceCaps(hDC, LOGPIXELSY)
   ReleaseDC 0, hDC
End Sub

Private Sub Calendar1_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Long, ByVal Y As Long)
 With Calendar1
      Dim d As Date
      Twips2Pixels X, Y
      d = .DateFromPoint(X, Y)
      If Not (d = 0) Then
         MsgBox FormatDateTime(d)
      End If
   End With
End Sub
```

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,
AxEXCALENDARLib._ICalendarEvents_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:
AxEXCALENDARLib._ICalendarEvents_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
AxEXCALENDARLib._ICalendarEvents_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(oCalendar,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.Calendar.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Calendar.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	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

he MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. Use the DateFromPoint property to retrieve the date from the cursor.

Use the following sample to retrieves the date over the cursor:

```
Private Sub Calendar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim d As Date
   d = Calendar1.DateFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
   If d <> 0 Then
      Debug.Print FormatDateTime(d)
   End If
End Sub
```

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,
AxEXCALENDARLib._ICalendarEvents_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:
AxEXCALENDARLib._ICalendarEvents_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
AxEXCALENDARLib._ICalendarEvents_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(oCalendar,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.Calendar.1::OLE_XPOS_PIXELS,Y as
```

```
OLE::Exontrol.Calendar.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. Use the DateFromPoint property to get the date from cursor.

The following sample prints the date from the cursor:

```
Private Sub Calendar1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   With Calendar1
      Dim d As Date
      d = .DateFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
      If Not (d = 0) Then
         Debug.Print FormatDateTime(d)
      End If
   End With
End Sub
```

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,
AxEXCALENDARLib._ICalendarEvents_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:
AxEXCALENDARLib._ICalendarEvents_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
AxEXCALENDARLib._ICalendarEvents_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(oCalendar,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.Calendar.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Calendar.1::OLE_YPOS_PIXELS)
```

```
end function
```

dBASE

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

event RClick ()

Fired when right mouse button is clicked

Type	Description

Use the RClick event to provide a context menu for your control. Use the MouseUp event to provide a context menu to the control, if you require the cursor coordinates.

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(oCalendar)
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 SelectionChanged ()

Fired when the selection was changed.

Type	Description

Use the SelectionChanged event to notify your application when the selection was changed. Use the SelDate and SelectDate properties to find the selected date(s). Use the SelDate if the SingleSel is True. Use the SelectDate is the SingleSel is False. Use the FocusDate property to specify the date that has the focus.

The following sample shows how to print the selected date when the SingleSel is True:

```
Private Sub Calendar1_SelectionChanged()
   Debug.Print FormatDateTime(Calendar1.SelDate())
End Sub
```

The following sample prints the selected date(s) if the SingleSel is False, (the control accepts multiple selection):

```
Private Sub Calendar1_SelectionChanged()
   Dim i As Long
   For i = 0 To Calendar1.SelCount() - 1
      Debug.Print FormatDateTime(Calendar1.SelectDate(i))
   Next
End Sub
```

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

C#

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

VB

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

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

C#

```
private void SelectionChanged(object sender, EventArgs e)
{
```

```
}
```

C++

```
void OnSelectionChanged()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event SelectionChanged()
end event SelectionChanged
```

VB.NET

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

VB6

```
Private Sub SelectionChanged()
End Sub
```

VBA

```
Private Sub SelectionChanged()
End Sub
```

VFP

```
LPARAMETERS nop

```

Xbas…

```
PROCEDURE OnSelectionChanged(oCalendar)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComSelectionChanged
    Forward Send OnComSelectionChanged
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_SelectionChanged()
{
}
```

XBasic

```
function SelectionChanged as v ()
end function
```

dBASE

```
function nativeObject_SelectionChanged()
return
```

Expressions

An expression is a string which defines a formula or criteria, that's evaluated at runtime. The expression may be a combination of variables, constants, strings, dates and operators/functions. For instance 1000 format `` gets 1,000.00 for US format, while 1.000,00 is displayed for German format.

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.

Usage examples:

- 100 + 200, adds numbers and returns 300
- "100" + 200, concatenates the strings, and returns "100200"
- currency(1000) displays the value in currency format based on the current regional setting, such as "$1,000.00" for US format.
- 1000 format `` gets 1,000.00 for English format, while 1.000,00 is displayed for German format
- 1000 format `2|.|3|,` always gets 1,000.00 no matter of settings in the control panel.
- upper("string") converts the giving string in uppercase letters, such as "STRING"
- date(dateS('3/1/' + year(9:=#1/1/2018#)) + ((1:=(((255 - 11 * (year(=:9) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(=:9) + int(year(=:9) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7)) returns the date the Easter Sunday will fall, for year 2018. In this case the expression returns #4/1/2018#. If #1/1/2018# is replaced with #1/1/2019#, the expression returns #4/21/2019#.

Listed bellow are all predefined constants, operators and functions the general-expression supports:

The constants can be represented as:

- numbers in **decimal** format (where dot character specifies the decimal separator). For instance: -1, 100, 20.45, .99 and so on
- numbers in **hexa-decimal** format (preceded by **0x** or **0X** sequence), uses sixteen distinct symbols, most often the symbols 0-9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. Hexadecimal numerals are widely used by computer system designers and programmers. As each hexadecimal digit represents four binary digits (bits), it allows a more human-friendly representation of binary-coded values. For instance, 0xFF,

0x00FF00, and so so.
- **date-time** in format **#mm/dd/yyyy hh:mm:ss#**, For instance, #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM
- **string**, if it starts / ends with any of the **'** or ` or **"** characters. If you require the starting character inside the string, it should be escaped ( preceded by a \ character ). For instance, `Mihai`, "Filimon", 'has', "\"a quote\"", and so on

The predefined constants are:

- **bias** (BIAS constant), defines the difference, in minutes, between Coordinated Universal Time (UTC) and local time. For example, Middle European Time (MET, GMT+01:00) has a time zone bias of "-60" because it is one hour ahead of UTC. Pacific Standard Time (PST, GMT-08:00) has a time zone bias of "+480" because it is eight hours behind UTC. For instance, date(value - bias/24/60) converts the UTC time to local time, or date(date('now') + bias/24/60) converts the current local time to UTC time. For instance, "date(value - bias/24/60)" converts the value date-time from UTC to local time, while "date(value + bias/24/60)" converts the local-time to UTC time.
- **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 :

- a **MIN** b (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.
- a **MAX** b (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. The type operator may return any of the following: 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. For instance *type(%1) = 8* specifies the cells (on the column with the index 1) that contains string values.
- **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#
- **hex** (unary operator) converts the giving string from hexa-representation to a numeric value, or converts the giving numeric value to hexa-representation as string. For instance, hex(`FF`) returns 255, while the hex(255) or hex(0xFF) returns the `FF` string. The hex(hex(`FFFFFFFF`)) always returns `FFFFFFFF` string, as the second hex call converts the giving string to a number, and the first hex call converts the returned number to string representation (hexa-representation).

The bitwise operators for numbers are:

- a **bitand** b (binary operator) computes the AND operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitand 0x10111000 returns 0x00001000.
- a **bitor** b (binary operator) computes the OR operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitor 0x10111000 returns 0x11111000.
- a **bitxor** b (binary operator) computes the XOR (exclusive-OR) operation on bits of a and b, and returns the unsigned value. For instance, 0x01110010 bitxor 0x10101010 returns 0x11011000.
- a **bitshift** (b) (binary operator) shifts every bit of a value to the left if b is negative, or to the right if b is positive, for b times, and returns the unsigned value. For instance, 128 bitshift 1 returns 64 (dividing by 2) or 128 bitshift (-1) returns 256 (multiplying by

2)
- **bitnot** (unary operator) flips every bit of x, and returns the unsigned value. For instance, bitnot(0x00FF0000) returns 0xFF00FFFF.

The 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

The 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"
- a **startwith** b (binary operator) specifies whether a string starts with specified string (

0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- a **endwith** b (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- a **contains** b (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- a **left** b (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- a **right** b (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 (binary operator) 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.
- a **like** b (binary operator) compares the string a against the pattern b. The pattern b may contain wild-characters such as *, ?, # or [] and can have multiple patterns separated by space character. In order to have the space, or any other wild-character inside the pattern, it has to be escaped, or in other words it should be preceded by a \ character. For instance value like `F*e` matches all strings that start with F and ends on e, or value like `a* b*` indicates any strings that start with a or b character.
- a **lpad** b (binary operator) pads the value of a to the left with b padding pattern. For instance, 12 lpad "0000" generates the string "0012".
- a **rpad** b (binary operator) pads the value of a to the right with b padding pattern. For instance, 12 lpad "____" generates the string "12__".
- a **concat** b (binary operator) concatenates the a (as string) for b times. For instance, "x" concat 5, generates the string "xxxxx".

The 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 expression supports also **immediate if** (similar with iif in visual basic, or ? : in C++) ie cond ? value_true : value_false, which means that once that cond is true the value_true is used, else the value_false is used. Also, it supports variables, up to 10 from 0 to 9. For instance, 0:="Abc" means that in the variable 0 is "Abc", and =:0 means retrieves the value of the variable 0. For instance, the len(%0) ? (0:=(%1+%2) ? currency(=:0) else `` ) : `` gets the sum between second and third column in currency format if it is not zero, and only if the first column is not empty. As you can see you can use the variables to avoid computing several times the same thing (in this case the sum %1 and %2 .

property CalendarCombo.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 / TemplatePut method 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.

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, 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.*